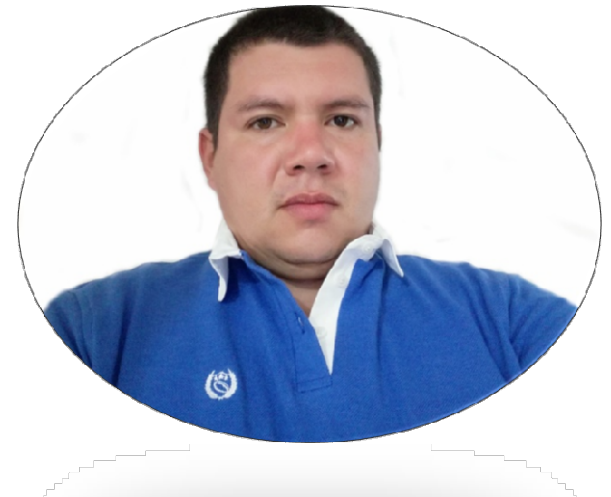


MikroTik Scripting

Introducción básica a la
programación de scripts con
MikroTik

MikroTik Scripting

- David Alejos.
- Email: david.alejos@outlook.com.
- Analista de Sistemas, egresado de UCLA.
- Certificaciones MikroTik: MTCNA, MTCRE, MTCTCE.
- Github: github.com/dalejos



Temas .

- Que es un algoritmo y sus características.
- Lenguajes de programación.
- Clasificación de los lenguajes por paradigma.
- Elementos básicos de un lenguaje de programación.
- Por que aprende un lenguaje de scripting.
- Introducción a Mikrotik Scripting.

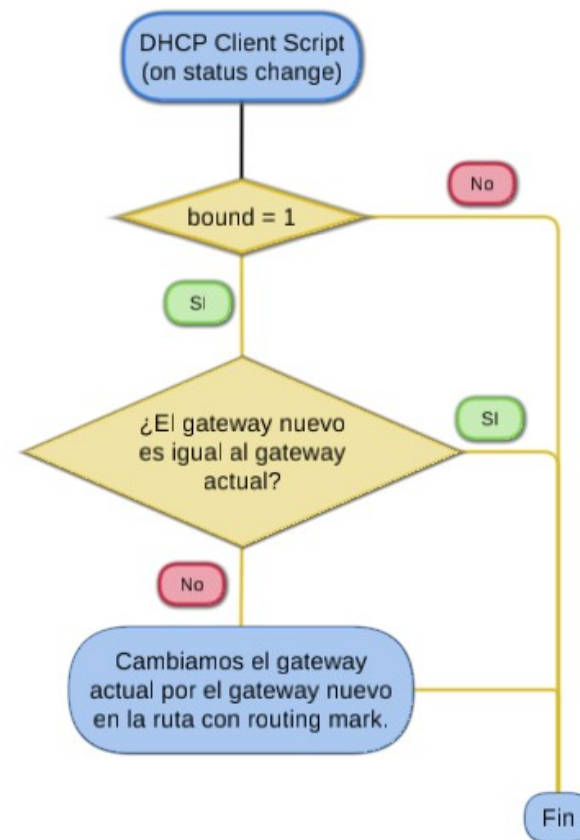


Temas .

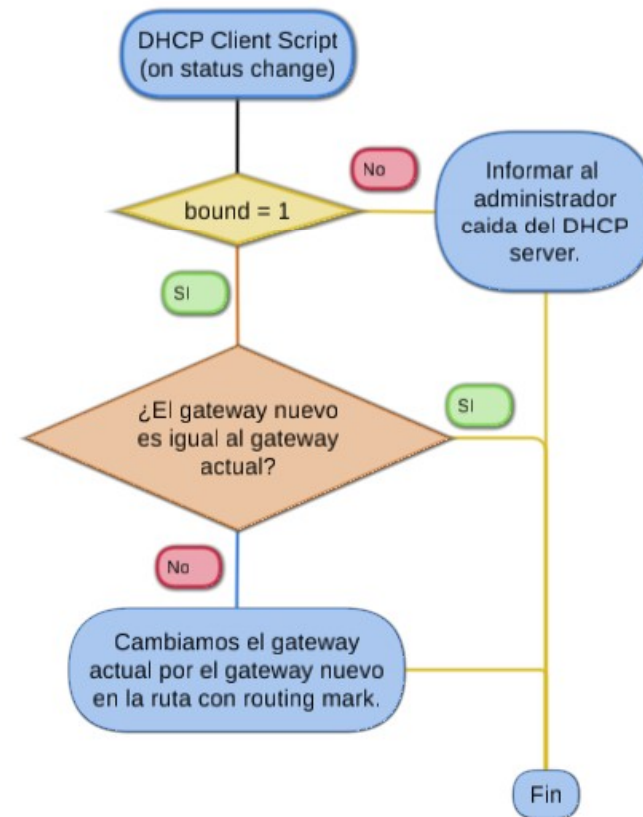
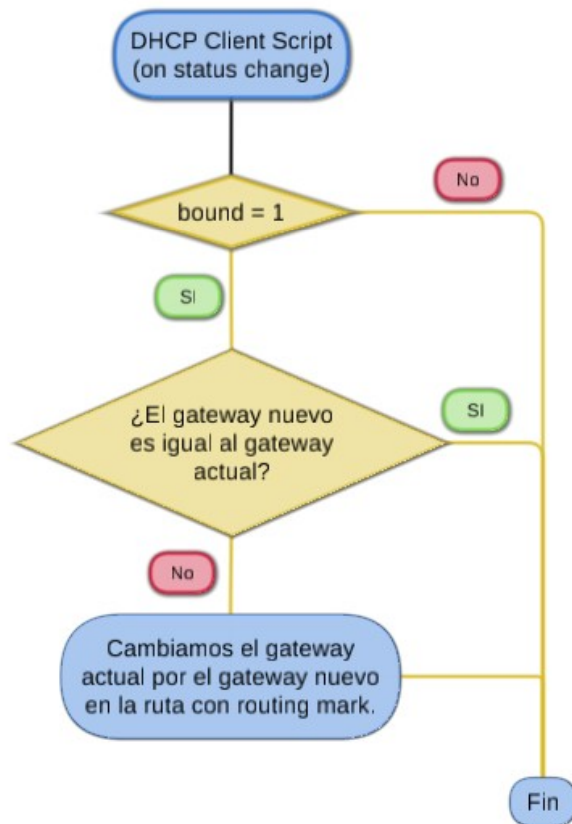
- Herramientas para MikroTik Scripting.
- Conociendo el lenguaje de Scripting en MikroTik.
- Laboratorio de scripting con MikroTik.

Que es un algoritmo y sus características.

- Conjunto de pasos, instrucciones o acciones que se deben seguir para resolver un problema.
- Existen una gran cantidad de algoritmos, hay que escoger el más efectivo.

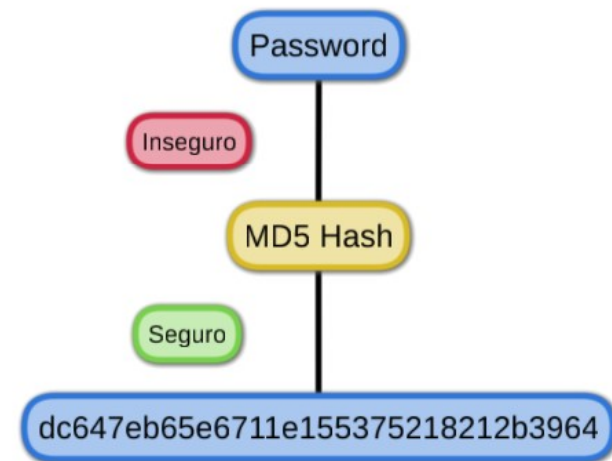


Que es un algoritmo y sus características.



Que es un algoritmo y sus características.

- Tiene que ser preciso.
- Tiene que estar bien definido.
- Tiene que ser finito.
- La programación es adaptar el algoritmo al ordenador.
- El algoritmo es independiente según donde lo implemente.



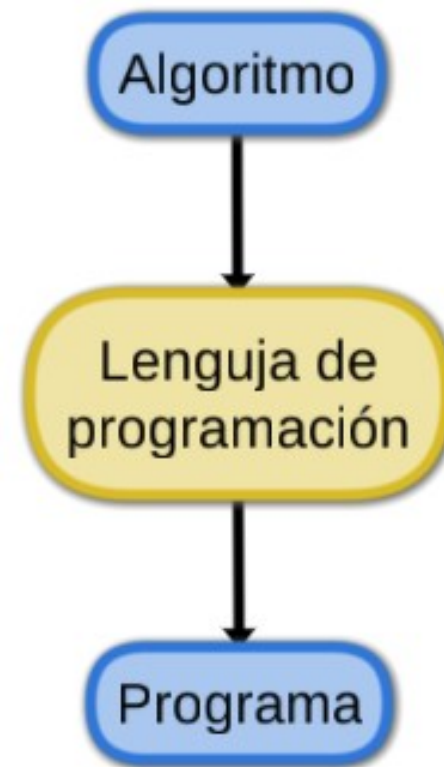
Lenguajes de programación.

- Es un lenguaje formal o artificial, es decir, un lenguaje con reglas gramaticales bien definidas.



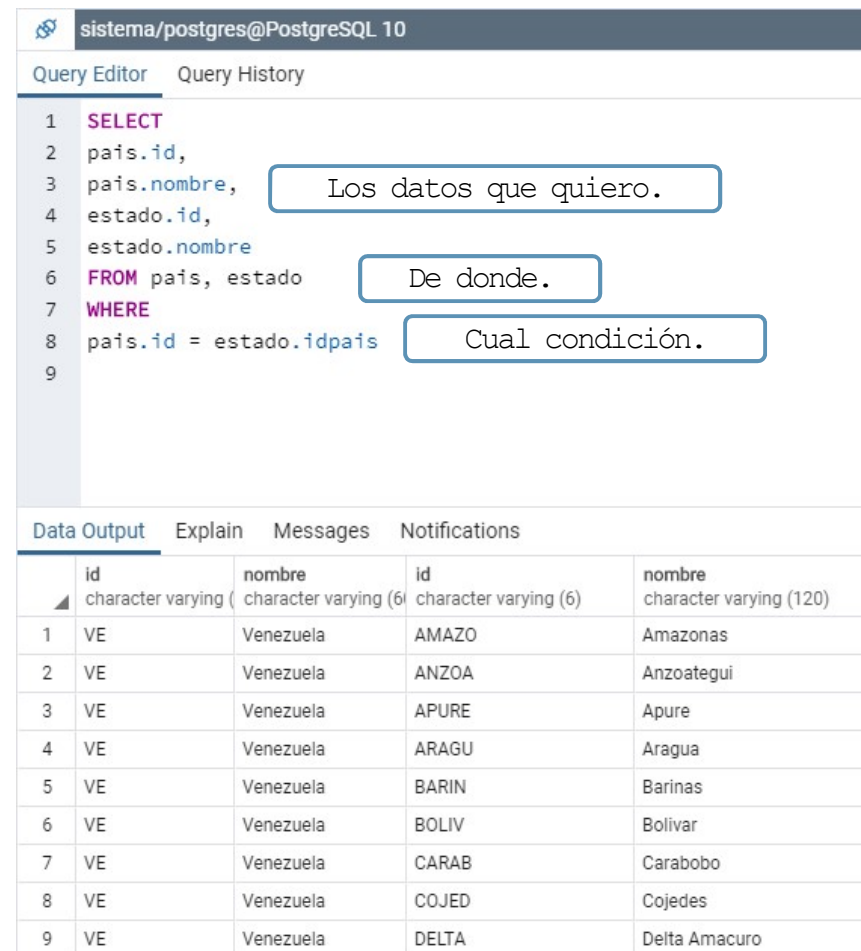
Lenguajes de programación.

- Proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico y/o lógico de una computadora, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas.



Clasificación de los lenguajes por paradigma.

- En general, la mayoría de paradigmas son variantes de los dos tipos principales de programación, declarativa e **imperativa**.
- En la programación declarativa las sentencias que se utilizan lo que hacen es describir el problema que se quiere solucionar



The screenshot shows a PostgreSQL Query Editor window titled 'sistema/postgres@PostgreSQL 10'. The 'Query Editor' tab is active, displaying a SQL query with line numbers 1 through 9. The query is: `SELECT pais.id, pais.nombre, estado.id, estado.nombre FROM pais, estado WHERE pais.id = estado.idpais`. Three blue boxes with white text provide annotations: 'Los datos que quiero.' points to the SELECT clause, 'De donde.' points to the FROM clause, and 'Cual condición.' points to the WHERE clause. Below the query editor, the 'Data Output' tab is active, showing a table with 4 columns: 'id', 'nombre', 'id', and 'nombre'. The table contains 9 rows of data, representing a join of the 'pais' and 'estado' tables.

	id character varying (6)	nombre character varying (60)	id character varying (6)	nombre character varying (120)
1	VE	Venezuela	AMAZO	Amazonas
2	VE	Venezuela	ANZOA	Anzoategui
3	VE	Venezuela	APURE	Apure
4	VE	Venezuela	ARAGU	Aragua
5	VE	Venezuela	BARIN	Barinas
6	VE	Venezuela	BOLIV	Bolivar
7	VE	Venezuela	CARAB	Carabobo
8	VE	Venezuela	COJED	Cojedes
9	VE	Venezuela	DELTA	Delta Amacuro

Clasificación de los lenguajes por paradigma.

- En la **programación imperativa** se describe paso a paso un **conjunto de instrucciones** que deben **ejecutarse** para variar el estado del programa y **hallar la solución**, es decir, **un algoritmo** en el que se describen los **pasos necesarios** para **solucionar el problema**.

```
:local addressType;
:set addressType do={
  :local ipAddress [:toip $1];
  :if ([:len $ipAddress] = 0) do={
    :return "UNKNOWN";
  }
  :local isPrivate \
    ((10.0.0.0 = ($ipAddress&255.0.0.0)) \
    or (172.16.0.0 = ($ipAddress&255.240.0.0)) \
    or (192.168.0.0 = ($ipAddress&255.255.0.0)));
  :if ($isPrivate) do={
    :return "PRIVATE";
  }
  :local isReserved \
    ((0.0.0.0 = ($ipAddress&255.0.0.0)) \
    or (127.0.0.0 = ($ipAddress&255.0.0.0)) \
    or (169.254.0.0 = ($ipAddress&255.255.0.0)) \
    or (224.0.0.0 = ($ipAddress&240.0.0.0)) \
    or (240.0.0.0 = ($ipAddress&240.0.0.0)));
  :if ($isReserved) do={
    :return "RESERVED";
  }
  :return "PUBLIC";
}
```

Instrucción 1
Instrucción 2
Instrucción 3
Instrucción n

Clasificación de los lenguajes por paradigma.

- Programación **imperativa** o por procedimientos (Mikrotik).
- Programación orientada a objetos.
- Programación dinámica.
- Programación dirigida por eventos.
- Programación declarativa.
- Programación funcional.
- Programación multiparadigma.
- Programación reactiva.
- Lenguaje específico del dominio o DSL. (Mikrotik)

Elementos básicos de un lenguaje de programación.

- Delimitadores.
- Tipos de datos.
- Palabras reservadas.
- Sentencias.
- Bloques de código.
- Comentarios.
- Funciones.
- Bucles.
- Operadores.

```
# Palabras reservadas

and
or
in
menu item properties

#Delimitadore
() [] {} : ; $ /

#Tipos de datos

num (number) - 64bit signed integer, (9.223.372.036.854.775.807),
               formato hexadecimal. Ejemplo 32, 0xA
bool (boolean) - true o false.
str (string) - secuencia de caracteres. Ejemplo "Esto es un string"
ip - IP address. Ejemplo 192.168.1.1
ip-prefix - IP prefix. Ejemplo 192.168.1.0/24
ip6 - IPv6 address. Ejemplo 2001:0db8:85a3:0000:0000:8a2e:0370:7334
ip6-prefix - IPv6 prefix. Ejemplo 2001:0db8:85a3:0000:0000:0000:0000/64
id (internal ID) - Id de cada item en un menu, hexadecimal, prefijo "*", secuencial
time - Fecha u hora. Ejemplo apr/29/2020, 01:00:00
array - Secuencia de valores en un arreglo. Ejemplo {1;2;3},
        {"token"<="XXXXX-XXX";"id"=3598756}
nil - Tipo de dato asignado a una variable sin valor especificado.

#Operadores

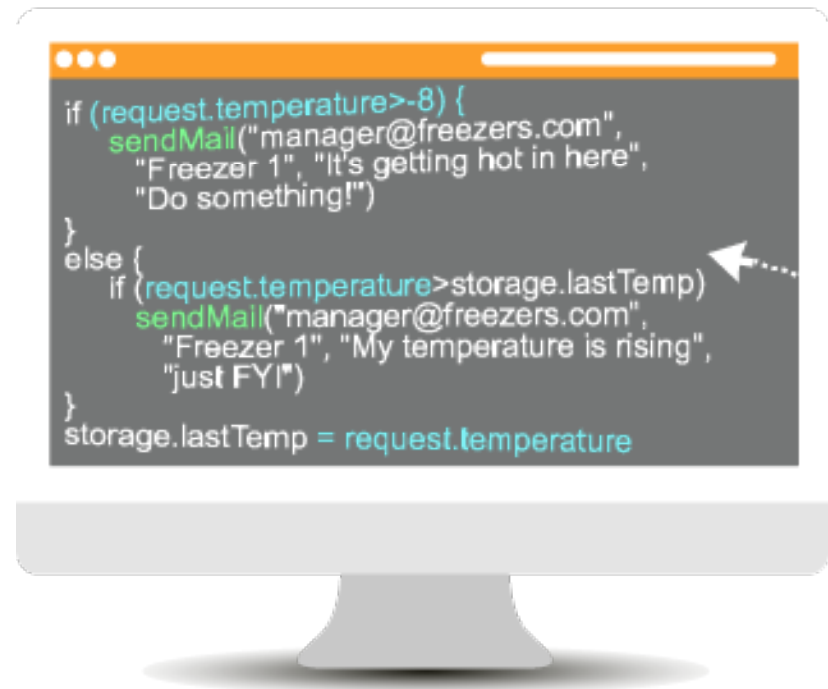
Aritmeticos + - * / %
Relacional < > = <= >= !=
Logicos ! && and || or in
Operadores
bit a bit ~ | ^ & << >>
Operadores de
concatenación . ,
Otros
operadores [] () $ ~ ->

#Comandos globales y operador global :

/          terminal error for if nothing put set tobool toip6 totime
:          delay execute foreach len parse resolve time toid tonum typeof
environment do find global local pick return toarray toip tostr while
```

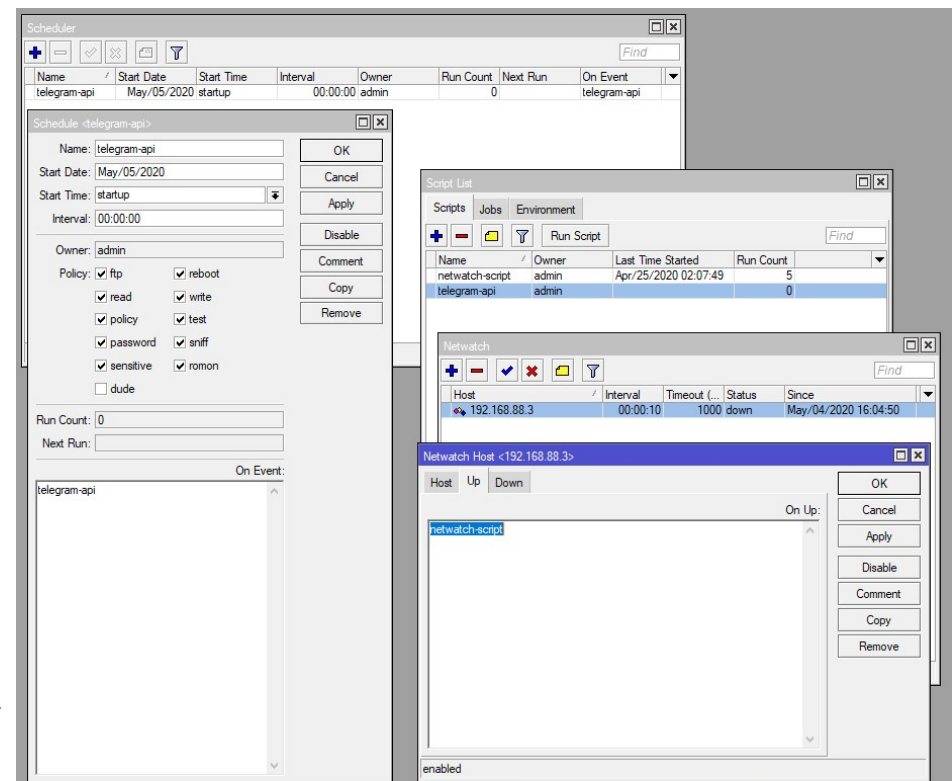
Por que aprende un lenguaje de scripting.

- Desarrolla nuestra lógica y manera de pensar.
- Mejora tu creatividad.
- Fácil de crear, modificar, ejecutar (no requieren compilar)
- Brindar soluciones Ad-hoc.
- Grades empresas usan lenguajes de scripting como forma de automatización de procesos.
- El auge cada vez mayor del IoT.

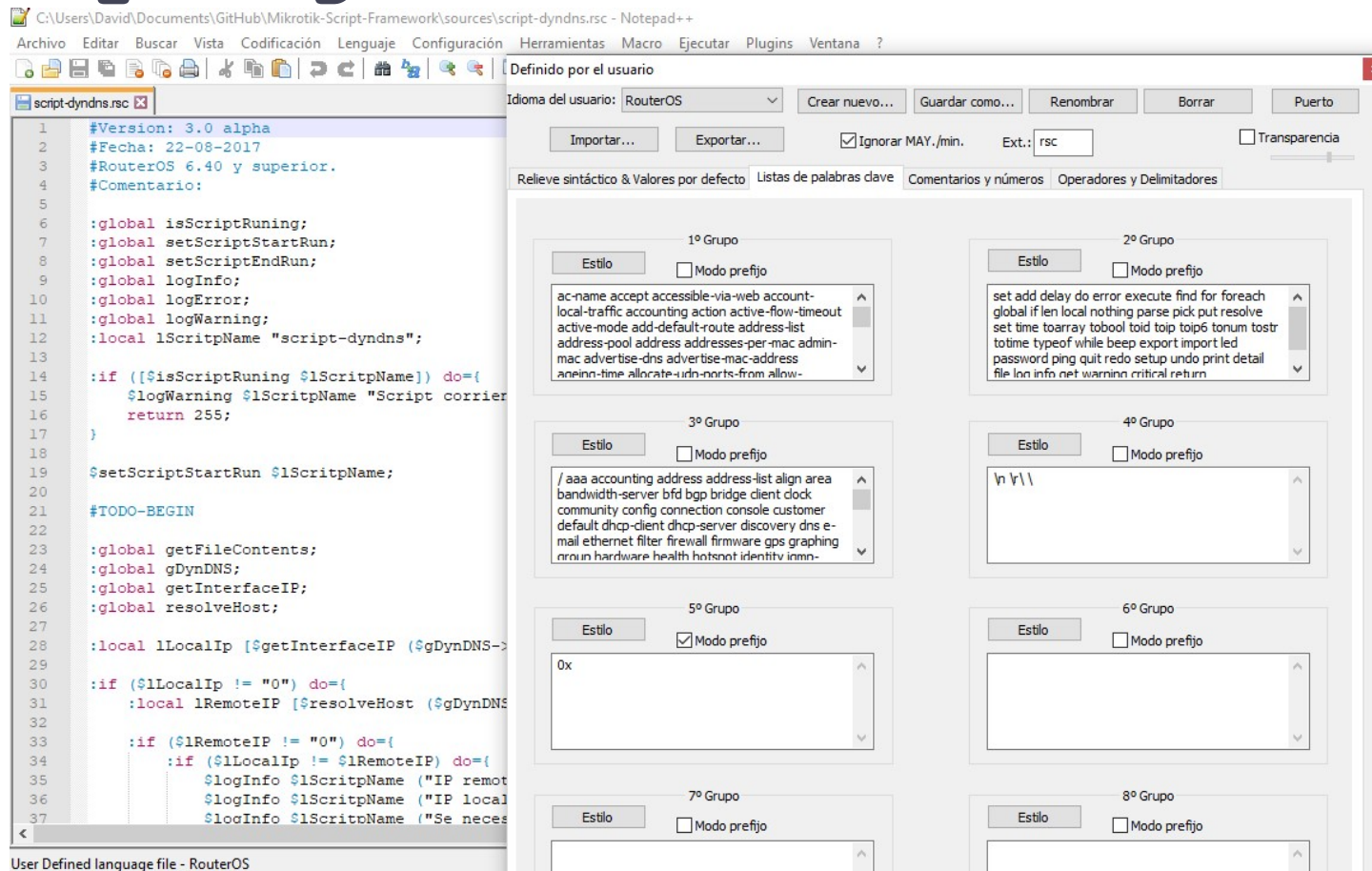


Introducción a Mikrotik Scripting.

- Lenguaje de script incorporado en RouterOS.
- Proporciona una forma de automatizar tareas de mantenimiento, recolección y análisis de datos e interactuar con otros sistemas.
- Se pueden escribir y ejecutar directamente en la consola ó el repositorio.
- Ejecución por eventos (Scheduler, Netwatch, DHCP, PPP, Hotspot)



Herramientas para Mikrotik Scripting.



Herramientas para Mikrotik Scripting.

REGEXPER

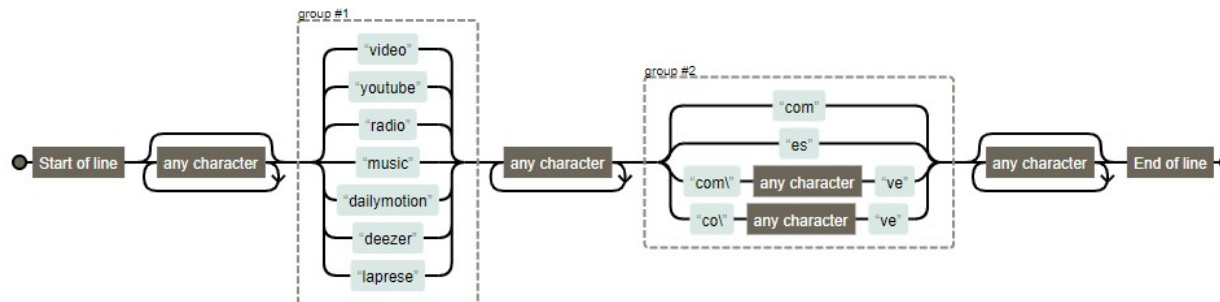
You thought you only had two problems...

- Changelog
- Documentation
- Source on GitLab

```
^.*(video|youtube|radio|music|dailymotion|deezer|laprese).+(com|es|com\\.ve|co\\.ve).*$
```

Display

[Download SVG](#) // [Download PNG](#) // [Permalink](#)



Herramientas para Mikrotik Scripting.



Anchors		Sample Patterns	
^	Start of line +	([A-Za-z0-9-]+)	Letters, numbers and hyphens
\A	Start of string +	(\d{1,2}\d{1,2}\d{4})	Date (e.g. 21/3/2006)
\$	End of line +	([^\s]+(?:\.(jpg gif png)))\.	jpg, gif or png image
\Z	End of string +	(^[1-9]{1}\$ ^[1-4]{1}[0-9]{1}\$ ^50\$)	Any number from 1 to 50 inclusive
\b	Word boundary +	(#[A-Fa-f0-9]{3}([A-Fa-f0-9]{3})?)	Valid hexadecimal colour code
\B	Not word boundary +	((?=[^\d])(?=[a-z])(?=[A-Z]).{8,15})	8 to 15 character string with at least one upper case letter, one lower case letter, and one digit (useful for passwords).
\<	Start of word	(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Email addresses
\>	End of word	(\<(/?[^\>]+)\>)	HTML Tags
Character Classes		Note <i>These patterns are intended for reference purposes and have not been extensively tested. Please use with caution and test thoroughly before use.</i>	
\c	Control character		
\s	White space		
\S	Not white space		
\d	Digit		
\D	Not digit		
\w	Word		
\W	Not word		
\xhh	Hexadecimal character hh		
\Oxxx	Octal character xxx		
POSIX Character Classes		Quantifiers	Ranges
[[:upper:]]	Upper case letters	*	0 or more +
[[:lower:]]	Lower case letters	*?	0 or more, ungreedy +
[[:alpha:]]	All letters	+	1 or more +
		+?	1 or more, ungreedy +
		?	0 or 1 +
		??	0 or 1, ungreedy +
		{3}	Exactly 3 +
		{3,}	3 or more +
		{3,5}	3, 4 or 5 +
		{3,5}?	3, 4 or 5, ungreedy +
		.	Any character except new line (\n) +
		(a b)	a or b +
		(...)	Group +
		(?:...)	Passive Group +
		[abc]	Range (a or b or c) +
		[^abc]	Not a or b or c +
		[a-q]	Letter between a and q +
		[A-Q]	Upper case letter + between A and Q +

Herramientas para Mikrotik Scripting.



[Main Page](#)
[Recent changes](#)

[Tools](#)
[What links here](#)
[Related changes](#)
[Special pages](#)
[Printable version](#)
[Permanent link](#)
[Page information](#)

[Manual](#)

[Discussion](#)

[Read](#)

[View source](#)

[View history](#)



Manual:Scripting

[Contents](#) [\[hide\]](#)

- 1 Scripting language manual
 - 1.1 Line structure
 - 1.1.1 Command line
 - 1.1.2 Physical Line
 - 1.1.3 Comments
 - 1.1.3.1 Example
 - 1.1.4 Line joining
 - 1.1.4.1 Example
 - 1.1.5 Whitespace between tokens
 - 1.1.6 Scopes
 - 1.1.6.1 Global scope
 - 1.1.6.2 Local scope
 - 1.2 Keywords
 - 1.3 Delimiters
 - 1.4 Data types
 - 1.4.1 Constant Escape Sequences
 - 1.4.1.1 Example
 - 1.5 Operators
 - 1.5.1 Arithmetic Operators

Applies to
RouterOS:
any



Conociendo el lenguaje de Scripting en MikroTik.

```
Terminal

MMM      MMM      KKK                      TTTTTTTTTT      KKK
MMM      MMM      KKK                      TTTTTTTTTT      KKK
MMM MMM MMM III KKK KKK RRRRRR      OOOOOO      TTT      III KKK KKK
MMM MM  MMM III KKKKK RRR RRR OOO OOO      TTT      III KKKKK
MMM      MMM III KKK KKK RRRRRR      OOO OOO      TTT      III KKK KKK
MMM      MMM III KKK KKK RRR RRR OOOOOO      TTT      III KKK KKK

MikroTik RouterOS 6.46.5 (c) 1999-2020      http://www.mikrotik.com/

[?]          Gives the list of available commands
command [?]  Gives help on the command and list of arguments

[Tab]        Completes the command/word. If the input is ambiguous,
              a second [Tab] gives possible options

/            Move up to base level
..           Move up one level
/command     Use command at the base level
[admin@Dude] > system identity
[admin@Dude] /system identity> :put [get]
name=Dude
[admin@Dude] /system identity> 
```