EXPRESIONES REGULARES, PATRONES, MÁSCARAS

Lee la explicación del enlace siguiente:

https://bioinf.comav.upv.es/courses/linux/unix/expresiones_regulares.html

Puedes consultar también:

https://enavas.blogspot.com.es/2008/04/el-shell-de-linux-comando-grep.html

http://francisconi.org/linux/expresiones-regulares

https://poesiabinaria.net/2015/10/9-trucos-para-manejar-cadenas-de-caracteres-en-bash-y-no-morir-en-el-intento/

http://www.vicente-navarro.com/blog/2007/04/13/expresiones-regulares-en-la-shell-ejemplos-de-uso-con-grep-awk-y-sed/

Ahora veamos algunos ejemplos con expresiones regulares:

-Control de la entrada de datos que sea numérica y no vacía.

```
#!/bin/bash
clear
esnumero=0
#Control entrada  numérica
read  -p "Introduzca su número:" num;
until [ ! -z "$num" ] && [ "$num" = "$esnumero" ];do
    esnumero='^-?[0-9]+([.][0-9])+?$'
    if [[ $num =~ $esnumero ]];then
       echo "SÃ, es un número positivo, negativo o decimal";
       exit 1;
    else
       echo "No es número"
       read -p "Por favor, introduzca un número:" num;
    fi
done
```

Analicemos ahora la expresión regular:

```
^-?[0-9]+([.][0-9])+?$
```

La vamos dividiendo en partes:

^-¿[0-9]+ Que empiece por un o ningún – y siga por una combinación de números - de uno o más números

([.][0-9])+ seguido de un punto o números

?$ Que termine por una o ninguna de la combinación anterior

-Control de la entrada de datos que sea cadena y no vacía.

```
#!/bin/bash
clear
#Control entrada cadena
read -p "Introduzca su nombre:" nombre;
#repite hasta que el nombre no sea vacio y sea valido
until [ ! -z "$nombre" ] && [ "$nombre" = "$valido" ];do
    #Si el nombre es numérico lo vuelve a pedir hasta que sea valido
    if [ "$nombre" = "$(echo "$nombre"|grep -E '[[:digit:]]')" ];then
         read -p "Por favor, vuelva a introducir su nombre:" nombre;
    else
         valido="$nombre";
         #es valido cuando tiene letras y caracteres especiales.
    fi
done

echo "Terminado, su nombre es: $nombre"
```

-Control de la entrada de datos cadena estricta (sin caracteres especiales, ni números)

```
#!/bin/bash
clear
#Control entrada cadena
escadena='^[a-zA-Z]+[a-zA-Z]+?$'
read  -p "Introduzca su cadena:" cad;
```

```
until [ ! -z "$cad" ] && [ "$cad" = "$escadena" ];do
 if [[ $cad =~ $escadena ]];then
   echo "SÃ, es una cadena (sin nÃºmero ninguno)";
   exit 1;
 else
   echo "No es una cadena"
   read -p "Por favor, introduzca una cadena:" cad;
 fi
done
```

Analicemos ahora la expresión regular:

^[a-zA-Z]+[a-zA-Z]+?$

La vamos dividiendo en partes:

^[a-zA-Z]+ Que empiece por una combinación de letras mayúsculas o minúsculas - de uno o más caracteres

[a-zA-Z]+?$ Que termine por una o ninguna combinación de letras mayúsculas o minúsculas - de uno o más caracteres

-Control de la entrada NIF. El NIF entra como parámetro del script

```
#!/bin/bash
#Ejecución del script:   ./ValidaNif nif. El nif escrito con 8 dígitos y una
letra
cadena="TRWAGMYFPDXBNJZSQVHLCKET"
#orden de letras que corresponden a cada dni
longitud=${#1}
               #Calcula la longitud del primer parámetro
if [ $longitud -eq 9 ];then
   pos=`expr ${1:0:8} % 23`
               #pos indica la posicion en cadena para saber la letra
   letra=${cadena:pos:1}
               #letra es la letra que corresponde según pos
   letraentrada=`echo ${1:8}|tr [a-z] [A-Z]`
               #Pasa de minúsculas a mayúsculas para la comparación correcta
   if [ "$letra" = "$letraentrada" ];then
     echo "Número nif correcto"
     exit 0
   else
     echo "Letra inválida, no se corresponde con su dni"
     exit 1
   fi
else echo "Longitud de nif errónea, debe tener 8 dígitos y 1 letra"
fi
```

- **Obtener la dirección MAC de la interfaz de red eth0 de nuestra máquina:**

$ ifconfig eth0 | grep -oiE '([0-9A-F]{2}:){5}[0-9A-F]{2}'

Sacamos la dirección MAC de la interfaz eth0 de nuestra máquina haciendo un:

ifconfig eth0

Y aplicando el filtro grep:

grep -oiE '([0-9A-F]{2}:){5}[0-9A-F]{2}'

Las opciones que he usado en grep son:

-o Indica que la salida del comando debe contener sólo el texto que coincide con el patrón, en lugar de toda la línea,

como es lo habitual.
-i Lo he usado para que ignore la distinción entre mayúsculas y minúsculas.
-E Indica que vamos a usar una expresión regular extendida.

En cuanto a la expresión regular, podemos dividirla en dos partes:
([0-9A-F]{2}:){5} Buscamos 5 conjuntos de 2 carateres seguidos de dos puntos
[0-9A-F]{2} seguido por un conjunto de dos caracteres.

Como las direcciones MAC se representan en hexadecimal, los caracteres que buscamos son los números del 0 al 9 y las letras desde la A a la F.

- Extraer la lista de direcciones de correo electrónico de un archivo:
grep -Eio '[a-z0-9._-]+@[a-z0-9.-]+[a-z]{2,4}' fichero.txt

Utilizo las mismas opciones que en el caso anterior:

-o Indica que la salida del comando debe contener sólo el texto que coincide con el patrón, en lugar de toda la línea, como es lo habitual.
-i Lo he usado para que ignore la distinción entre mayúsculas y minúsculas.
-E Indica que vamos a usar una expresión regular extendida.

Analicemos ahora la expresión regular:
[a-z0-9._-]+@[a-z0-9.-]+[a-z]{2,4}

Al igual que antes, la vamos dividiendo en partes:
[a-z0-9._-]+ Una combinación de letras, números, y/o los símbolos . _ y - de uno o más caracteres
@ seguido de una arroba
[a-z0-9.-]+ seguido de una cadena de letras, números y/o los símbolos . y -
[a-z]{2,4} seguido de una cadena de entre dos y cuatro caracteres.

- Obtener la dirección IP de la interfaz de red eth1 de nuestra máquina:
$ ifconfig eth1 | grep -oiE '([0-9]{1,3}\.){3}[0-9]{1,3}' | grep -v 255

En el ejemplo anterior, hemos tomado la información que nos ofrece ifconfig:
ifconfig eth1

Hemos filtrado dicha información con el comando grep, obteniendo todas las direcciones IP que aparecen:
grep -oiE '([0-9]{1,3}\.){3}[0-9]{1,3}'

Por último, hemos filtrado la salida del comando anterior, para eliminar la dirección de broadcast junto con la máscara de red para quedarnos sólo con la dirección IP de la máquina:
grep -v 255

La línea anterior no mostraría las líneas que no contengan el valor 255, es decir, las direcciones de broadcast y máscara de red.

Analicemos ahora el comando grep:
grep -oiE '([0-9]{1,3}\.){3}[0-9]{1,3}'

Al igual que en los otros dos ejemplos de expresiones regulares uso las opciones -oiE en el comando grep:

-o Indica que la salida del comando debe contener sólo el texto que coincide con el patrón, en lugar de toda la línea, como es lo habitual.
-i Lo he usado para que ignore la distinción entre mayúsculas y minúsculas.

-E Indica que vamos a usar una expresión regular extendida.

En cuanto a la expresión regular:

'([0-9]{1,3}\.){3}[0-9]{1,3}'

([0-9]{1,3}\.){3} Representa 3 bloques de entre uno y tres dígitos separados por puntos. Observemos que como el punto es un metacaracter, tengo que usar el caracter de escape \ para que no sea interpretado como un metacaracter, sino como un caracter normal.

[0-9]{1,3} Representa el último bloque de la dirección IP, que está formado por un número de entre 1 y 3 dígitos