Programación con Shell Scripting: Sesión 9

Máster Universitario en Bioinformática

Universidad Internacional de Valencia

Dra. Paula Soler Vila
paula.solerv@professor.universidadviu.com

De:
Planeta Formación y Universidades

Aspectos a tratar

1. Trabajando con variables *(diapositivas en la sesión 8)*
2. Expresiones regulares
3. Comandos esenciales para el procesamiento de datos:
   - **grep**
   - **sed**
   - **awk**

¿Qué es una Expresión Regular (*regex*)?

Conjunto de caracteres especiales que permiten seleccionar o encontrar patrones en un texto

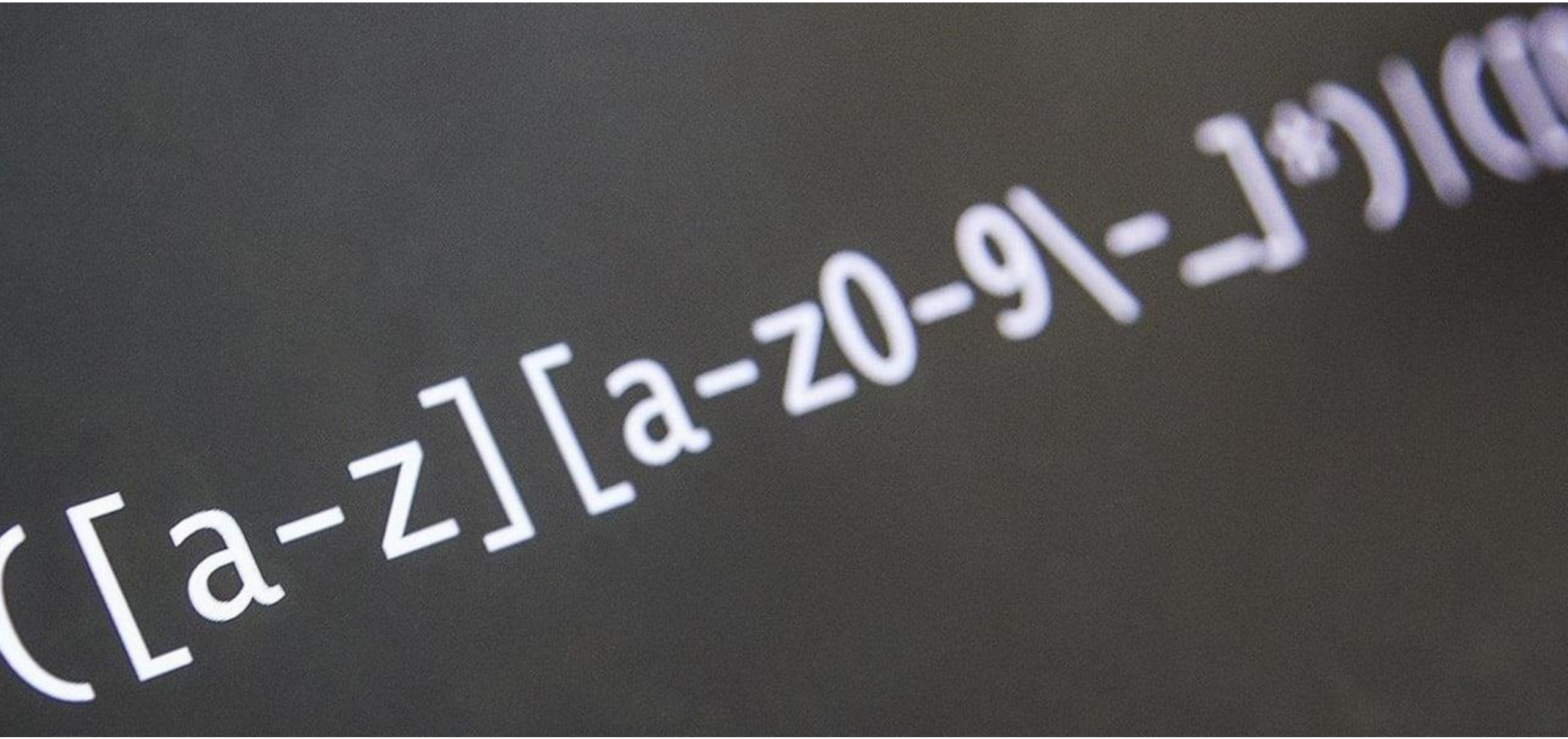
↓

Deseamos buscar la palabra «**linux**» en un texto

↓

Pero, ¿y si queremos buscar **todos los números** que aparecen en un texto? ¿O todas **las líneas que empiezan por una letra mayúscula**?

La solución es usar una expresión regular


([a-z][a-z0-9\-_]*)

Construcción de expresiones regulares

caracteres + metacaracteres = regex

Un **metacarácter** es un carácter escrito que tiene un significado especial para el ***shell***

*Patrón = aba** → **No** se representan a ellos mismos

Los metacaracteres más usados son: **. * ? + [ ] ( ) { } ^ $ | **

Escape (\)

A veces necesitamos **incluir** en nuestro patrón algún **metacarácter** como signo **literal**, es decir, por sí mismo y no por lo que representa.

Para indicar esto usamos como carácter de escape a la **barra invertida **

Expresión	Coincidencia en el texto
\$12	El precio del menú es de **$12**,50

Metacaracteres

Metacaracter de posicionamento o anclaje ^ y $

Los signos **^** y **$** sirven para indicar donde debe estar situado el patrón dentro de la cadena para considerar que existe una coincidencia.

- **^** indica que el patrón a buscar debe aparecer al **inicio** de la línea o cadena.

- **$** indica que el patrón a buscar debe aparecer al **final**, es decir antes de un carácter de nueva línea (\n).

Metacaracter de posicionamento o anclaje ^ y $

Expresión	Coincidencia en el texto
^el	**el** signo ^ indica que el patrón debe estar al inicio de la cadena o línea
$ila	Nos dijeron que formáramos una fila y nadie se puso en f**ila**
^$	se puede utilizar para encontrar líneas vacías, donde el inicio de una línea es inmediatamente seguido por el final de ésta

Comodín .

El metacarácter **.** (punto) es un **comodín** y representa **cualquier carácter**

incluso caracteres en blanco

Expresión	Coincidencia en el texto
.l	**El** precio d**el** menú es de $12,50, así que aproveche la oferta d**el** día (Noten en este ejemplo que también incluye la ocurrencia de un espacio en blanco seguido de la letra l)

Rango de Caracteres [ ]

Los **corchetes [ ]** definen una **clase de caracteres** y permiten encontrar cualquiera de los caracteres dentro del grupo especificado

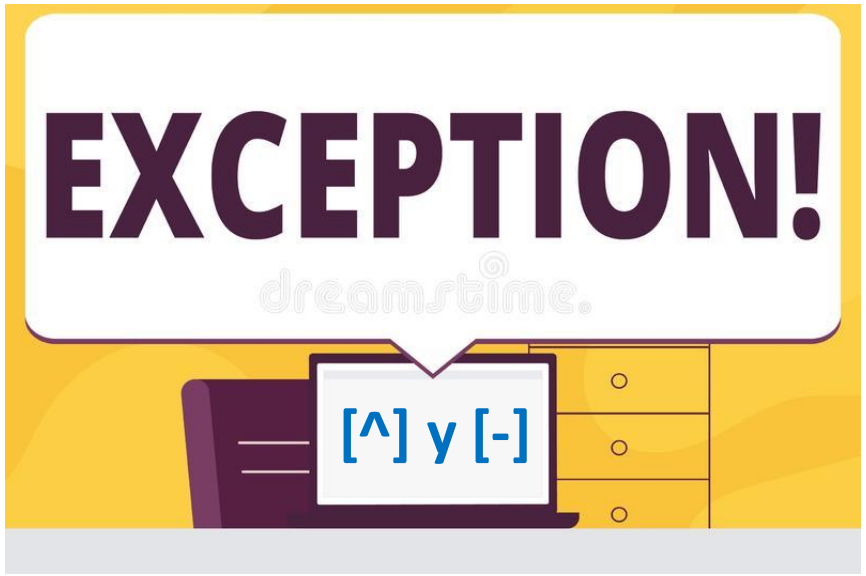
Queremos encontrar la palabra «**niño**», pero también queremos encontrar en caso de que hayan escrito la palabra con **n** en lugar de **ñ**

La expresión regular sería **ni[ñn]o**

Rango de Caracteres [ ]

La **mayoría de los metacaracteres pierden** su significado al ser utilizados dentro de **clases de caracteres.**

La expresión **[a.]** se refiere literalmente a la letra **a** y al carácter **punto**



Rango de Caracteres (casos especiales)

El acento circunflejo ^

^ al ser utilizado al comienzo de una clase de caracteres significa **negación**

La expresión **[^a]** se refiere a cualquier cadena que **NO** contenga la letra a

La expresión **[a^]** se refiere literalmente a la letra **a** y al **carácter ^**

El guión -

El guion **-** al ser utilizado dentro de corchetes indica que se trata de un **rango**

Cualquier letra del alfabeto en minúscula

[a-z]

Rango de Caracteres

Expresión	Coincidencia en el texto
[abc]	El patrón coincide con la cadena si en esta hay una a, una b o una c
[a-c]	Equivalente al caso anterior
c[aeo]sa	Coincide con las palabras **casa**, **cesa**, **cosa**
[^abc]	El patrón coincide con la cadena si no hay ninguna a, b y c
[0-9]	Coincide con una cadena que contenga cualquiera de los dígitos
[^0-9]	**Coincide con una cadena que no contenga ningún dígito**

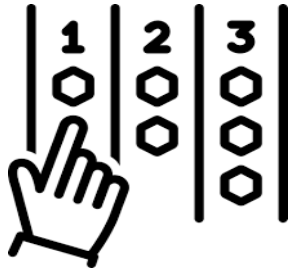
Clases de Caracteres

Examen

Clase	Caracteres	Significado
[:alnum:]	[A-Za-z0-9]	Caracteres alfanuméricos
[:word:]	[A-Za-z0-9_]	Caracteres alfanuméricos y _
[:alpha:]	[A-Za-z]	Caracteres alfabéticos
[:blank:]	[\t]	Espacio y tabulador
[:space:]	[\t\r\n\v\f]	Espacios
[:digit:]	[0-9]	Dígitos
[:lower:]	[a-z]	Letras minúsculas
[:upper:]	[A-Z]	Letras mayúsculas
[:punct:]	[][!"#$%&'()*+,./:;<=>?@\^_`{|}~-]	Caracteres de puntuación

Con grep se usaría por ejemplo con doble [[ ]]

Cuantificadores

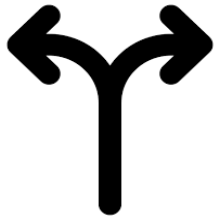


Indican **las veces** que deben **aparecer** los caracteres

Por defecto, se asume que el carácter debe aparecer **una vez**, pero podemos estar interesados en que **aparezca un número distinto** de veces

Cuantificador	Significado
?	El carácter aparece **cero** o **una** vez
*	El carácter aparece **cero**, **una** o **varias** veces
+	El carácter aparece al **menos una** vez
{n}	El carácter aparece **n** veces
{n,m}	El carácter aparece entre **n** y **m** veces

Expresiones Alternativas



La **barra vertical** se utiliza para **alternativas** que son evaluadas de izquierda a derecha

Valencia|Madrid -> La barra vertical (|) separa las expresiones alternativas

La palabra encontrada puede ser **Valencia o Madrid**

Ejemplos

Expresiones Regulares

^a

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "^a" texto.txt
aaTTGG
```

AT+G

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "AT+G" texto.txt
ATGC
AAATTTTGGGC
```

[0-9]

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "[0-9]" texto.txt
T333T
4AAA
```

A(TT)+G

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "A(TT)+G" texto.txt
AAATTTTGGGC
```

A(TT)+G indica que busque el conjunto TT seguido de la A

A{1,3}T

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "A{1,3}T" texto.txt
ATGC
AAATTTTGGGC
```

[:upper:]

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "[[:upper:]]" texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

Con el comando grep estas clases predefinidas hay que ponerlas con doble [[ ]] si no da error

Aspectos a tratar

1. Trabajando con variables
2. Expresiones regulares
3. Comandos esenciales para el procesamiento de datos:
   - **grep**
   - **sed**
   - **awk**

Comando *grep*

grep (g/re/p, ***g****lobally search for a* ***r****egular* ***e****xpression and* ***p****rint matching lines*)

Poderosa herramienta de búsqueda de regex en archivos de texto

La sintaxis general es:

```
grep PATRON ARCHIVO
```

Normalmente, los patrones se ponen entre comillas si contienen espacios o caracteres especiales.

Se recomienda usarlo con comillas dobles

Comando *grep: algunas opciones*

Argumento	Función
-v	Invierte la coincidencia o encuentra líneas que NO contengan el patrón
--color	Colorea el texto que coincide para facilitar su visualización
-F	Interpreta el patrón como una cadena literal
-H ,-h	Imprime o no imprime el nombre de archivo que coincida
-i	Ignora casos que coincidan con el patrón
-l	Lista los nombres de archivo que contienen el patrón (en lugar de coincidir)
-n	Imprime el número de líneas que contienen el patrón (en lugar de coincidir)
-c	Cuenta el número de coincidencias para un patrón
-o	Solo imprime el patrón que coincide
-w	Fuerza a que el patrón coincida con una palabra entera
-x	Fuerza patrones para que coincidan con toda la línea
-f	Obtiene patrones desde un archivo
-E	Interpreta el patrón como una expresión regular extendida

Algunas opciones del comando grep: archivo TSV

seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33
seq2 chr1 exp-HT33
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1

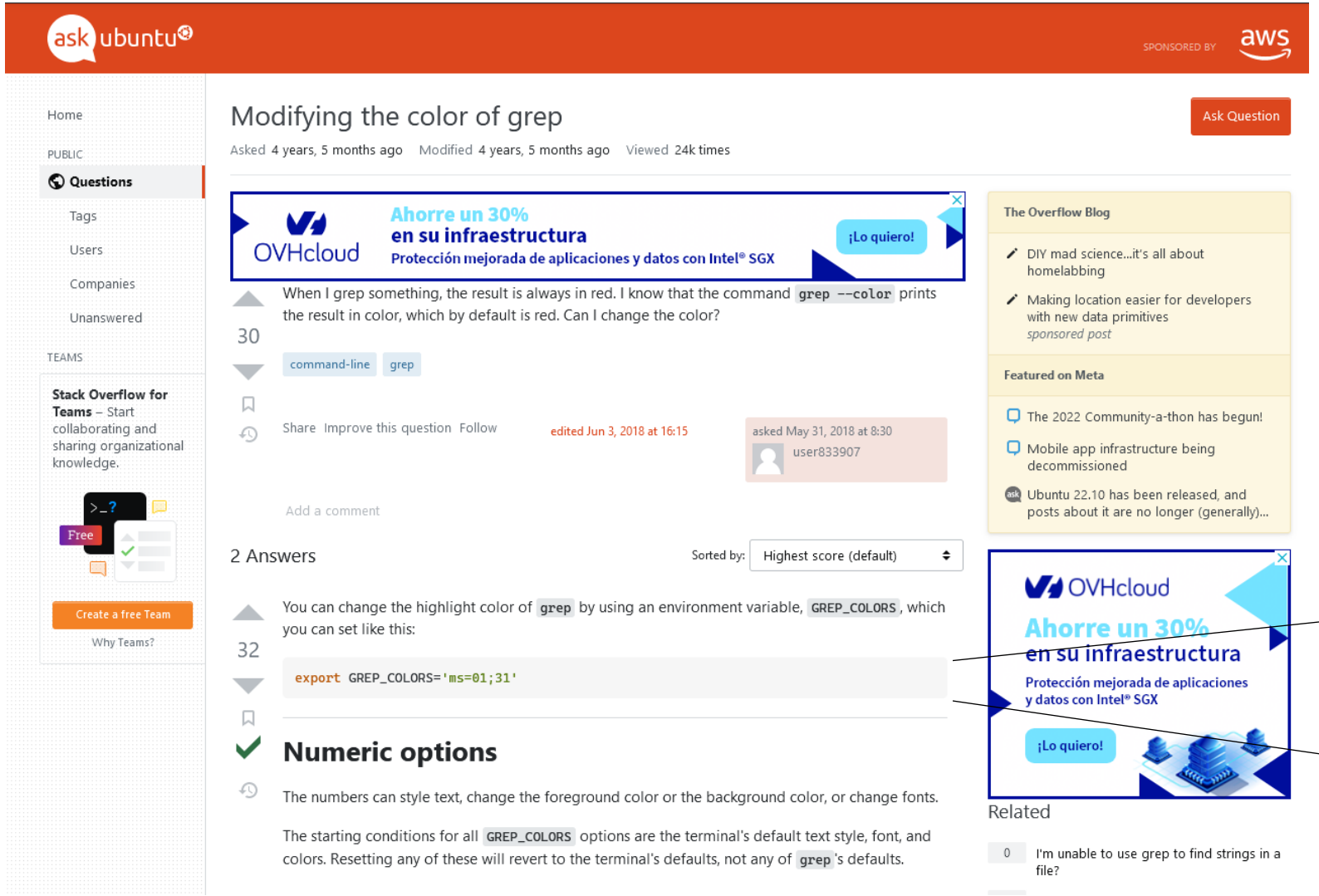
references.tsv

grep "seq1" references.tsv

seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33

Variable de entorno global
GREP_COLORS.

Modificando la variable global GREP_COLORS




```
export GREP_COLORS='ms=01;31'
```

https://askubuntu.com/questions/1042234/modifying-the-color-of-grep

Algunas opciones del comando grep: -n -o

```
seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33
seq2 chr1 exp-HT33
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
```

references.tsv

grep **-n** "seq1" references.txt →

```
1:seq1 chr1 exp-AC02
2:seq1 CHR1 exp-CC33
```

grep **-no** "seq1" references.txt →

```
1:seq1
2:seq1
```

Algunas opciones del comando grep: -c

-c

(cuenta en número de líneas coincidentes para un patrón dado)

seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33
seq2 chr1 exp-HT33
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1

references.tsv

grep -c "seq1" references.txt → **2**

grep -c "chr1" references.txt → **5**

Algunas opciones del comando grep: -c

```
seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33
seq2 chr1 exp-HT33
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
```

references.tsv

grep "1" references.txt

```
seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33
seq2 chr1 exp-HT33
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
```

grep -c "1" references.txt

9

Ojo que reporta lineas coincidentes, en cuantas lineas aparece el 1, no cuantas veces aparece el 1

Algunas opciones del comando grep: -v

seq1 chr1 exp-AC02
seq1 CHR1 exp-CC33
seq2 chr1 exp-HT33
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq5 chr12 exp-0011
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1

references.tsv

grep –v "chr1" references.txt

seq1 CHR1 exp-CC33
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq3 chr2 exp-FG90
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1
seq7 chr8 exp-0TT1

Algunas opciones del comando grep: -f

Lista_principal

SBP1
SIX5
PRDM1
SOX13
BAF1
TCF4
PAX5
CTCF
SOX133

Lista_secundaria

SOX13
BAF1
TCF4
PAX5

grep -f lista_secundaria.txt lista_principal.txt

SBP1
SIX5
PRDM1
SOX13
BAF1
TCF4
PAX5
CTCF
SOX133

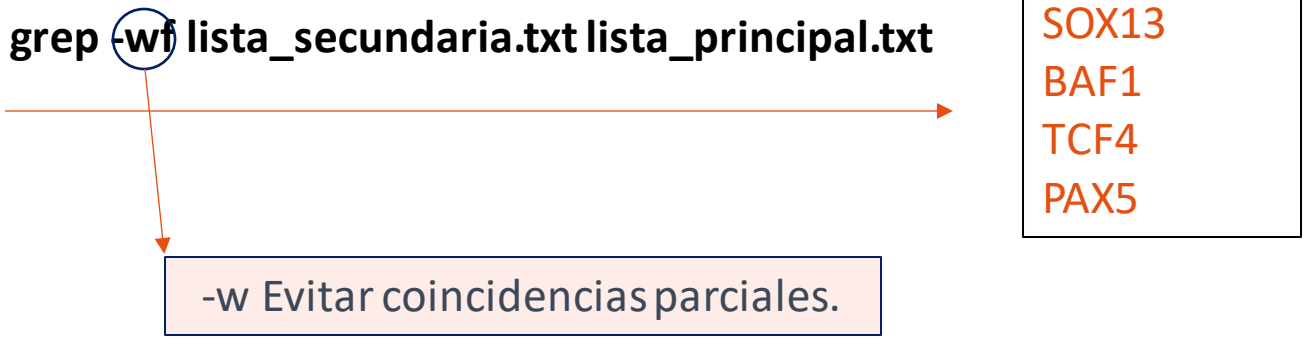
Algunas opciones del comando grep: -f

Lista_principal

SBP1
SIX5
PRDM1
SOX13
BAF1
TCF4
PAX5
CTCF
SOX133

Lista_secundaria

SOX13
BAF1
TCF4
PAX5



SOX13
BAF1
TCF4
PAX5

Diferentes versiones de grep: -G -E -F -P

Versión	Descripción
-G o --basic-regexp	Interpreta los patrones como regex básicas (BREs). Es la versión estándar de la herramienta
-E o --extended-regexp	**Interpreta los patrones como regex extendidas (EREs)**
-F o --fixed-strings	Interpreta los patrones como string fijos, no como regex
-P o --perl-regexp	Interpretar los patrones como expresiones regulares compatibles con Perl (PCREs)

Diferentes versiones de grep: -G -E -F -P

```
$ cat texto.txt
ATGC
AAATTTTGGGC
aaTTGG
T333T
4AAA
```

```
$ grep -E "AT+G" texto.txt
ATGC
AAATTTTGGC

$ grep "AT+G" texto.txt
$ grep "AT\+G" texto.txt
ATGC
AAATTTTGGC
```

Si no pones el \ lo entiende literalmente. Esto para son algunos caracteres como con el +, asi que mejor ponerlo siempre

Comando grep: Ejercicios

1.Buscar las líneas en las que aparece la palabra bash y la palabra root en el archivo /etc/passwd.

grep –n -e 'bash' -e 'shell' /etc/passwd

El normal: grep -n -E “bash” /etc/passwd

Un ejemplo para que te las reporte a la vez: grep -n -E “bash|root” /etc/passwd

Con -e concatenamos búsquedas también: grep -n -e “bash” -e “root” /etc/passwd

2.Buscar en el archivo /etc/group todas las líneas que empiezan por *m y* por *n*.

grep '^[mn]' /etc/group

3.En el fichero anterior imprimir todas las líneas que no empiezan por *m.*

grep -v '^m' /etc/group

4. ¿Qué ficheros o directorios en /etc contienen un número al final de su nombre?

ls /etc/ | grep '[0-9]$'

Comando grep: Archivo multifasta

```
$ cat multifasta.txt
>test1
ATAGATAGTAGTA
>test2
GGGGTTTTTT
>test3
AAAAAAAAAAA
```

Indica que se deben mostrar también la línea coincidente y la línea siguiente después de la coincidencia.

```
$ grep -E -A1 "^>test1" multifasta.txt
>test1
ATAGATAGTAGTA
```

A de after, también se puede usar con B de before. Y en vez de 1 puedes poner n lineas coincidentes.

Comando grep: Enzimas de restricción

Enzima	Sitio de reconocimiento
EcoRI	GAATTC
BfuI	GTATCC
FokI	GGATG
HincII	GTYRAC
HphI	GGTGA
PaeI	GCATGC
PdiI	GCCGGC
SmaI	CCCGGG
TaiI	ACGT
XhoI	CTCGAG

La enzima **HincII** presenta una secuencia de reconocimiento “degenerada”, en la que la letra Y puede ser los nucleótidos C o T y la letra R puede ser G o A.

Comando grep: Enzimas de restricción

Enzima	Sitio de reconocimiento
EcoRI	GAATTC
BfuI	GTATCC
FokI	GGATG
HincII	GTYRAC

La enzima **HincII** presenta una secuencia de reconocimiento "degenerada", en la que la letra Y puede ser los nucleótidos C o T y la letra R puede ser G o A.

¿Cómo buscaríais todos los sitios de reconocimiento posible en un archivo FASTA?

Comando grep: Enzimas de restricción

Enzima	Sitio de reconocimiento
EcoRI	GAATTC
BfuI	GTATCC
FokI	GGATG
HincII	GTYRAC

La enzima **HincII** presenta una secuencia de reconocimiento "degenerada", en la que la letra Y puede ser los nucleótidos C o T y la letra R puede ser G o A.

¿Cómo buscaríais todos los sitios de reconocimiento posible en un archivo FASTA?

 # Opción 1

```
grep -E
"GTCAAC|GTCGAC|GTTAAC|GTTGAC"
*.fasta
```

 # Opción 2

```
grep -E  "GT[CT][AG]AC" *.fasta
```

Universidad
Internacional
de Valencia

universidadviu.com

De:
Planeta Formación y Universidades