Programación con Shell Scripting: Sesión 10

Máster Universitario en Bioinformática

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

De:
Planeta Formación y Universidades

Aspectos a tratar

1. Comandos para el procesamiento de datos:
   - **grep**
   - **sed**
   - **awk**

Comando grep: Archivo multifasta

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

1) Identificación de una secuencia de interés

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

2) Comparación de ficheros con extracción inversa de coincidencias totales

```
$ cat sequence_remove.txt
>test1
ATAGATAGTAGTA

$ grep -v -w -f sequence_remove.txt multifasta.txt
>test2
GGGGTTTTTT
>test3
AAAAAAAAAAA
```

Comando sed *(stream editor)*

Editor de flujo de texto que se utiliza para realizar transformaciones de texto

1. **No** realiza ningún cambio directamente en el archivo.
2. Opera línea a línea
3. Búsqueda y sustitución
4. Da soporte a las expresiones regulares

Sintaxis básica

sed [opción(es)] ‘orden(es)’ archivo(s)

Se especifican mediante un guión – y la opción correspondiente con una letra (-e , -f ,-n ,-i,…)

Se especifican entre comillas simples

p: print

d: delete

i: insert

s: substitute

Sintaxis básica: Opciones

Parámetro	Explicación
-e	Hace que utilicen uno o varios scripts SED
-f	Hace que el script se extraiga de un archivo
-n	Los resultados no se deben emitir
-i	Crea un archivo temporal que posteriormente sustituye al archivo de origen
-u	No se utiliza ningún buffer de datos
-s	Varios archivos se tratan por separado en lugar de ser un único largo flujo de datos
-r	El comando acepta expresiones regulares ampliadas

Sintaxis básica: Órdenes

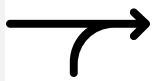
Orden	Descripción
a	append: añade a las líneas seleccionadas una o más líneas más
c	change: reemplaza las líneas seleccionadas por un nuevo contenido
d	delete: borra las líneas seleccionadas
g	get: copia el contenido del hold space al pattern space
G	GetNewline: añade el contenido del hold space al pattern space
h	hold: copia el contenido del pattern space al hold space
H	HoldNewLine: añade el contenido del pattern space al hold space
i	insert: inserta una o más líneas antes de las líneas seleccionadas
l	listing: muestra todos los caracteres no imprimibles
n	next: cambia a la siguiente orden de la línea siguiente del comando
p	print: muestra las líneas seleccionadas
q	quit: finaliza el comando SED de Linux
r	read: lee las líneas seleccionadas de un archivo
s	substitute: reemplaza una determinada cadena de caracteres por otra
x	xchange: intercambia el pattern space y el hold space entre sí
y	yank: sustituye un determinado carácter por otro
w	write: escribe líneas en el archivo de texto
!	Negation: aplica el comando a las líneas que no coinciden con la entrada.

Comando sed : ‘ ‘

```
$ cat dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

dias_de_la_semana.txt

```
sed ‘ ’ dias_de_la_semana.txt
```



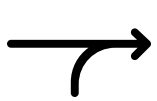
```
$ sed ' ' dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

Comando sed : ‘ ‘ -n (--quiet --silent)

```
$ cat dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

dias_de_la_semana.txt

```
sed ‘ ’ -n dias_de_la_semana.txt
```



```
$ sed ' ' -n dias_de_la_semana.txt
```

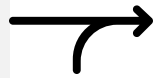
por defecto sed imprime de forma automática todas las lineas, pero con la opción -n va a suprimir esta impresión automática que tiene establecida por defecto

Comando sed : ‘p‘*(print)* -> Impresión específica de líneas de un archivo

```
$ cat dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

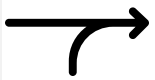
dias_de_la_semana.txt

sed ‘1p ’ dias_de_la_semana.txt



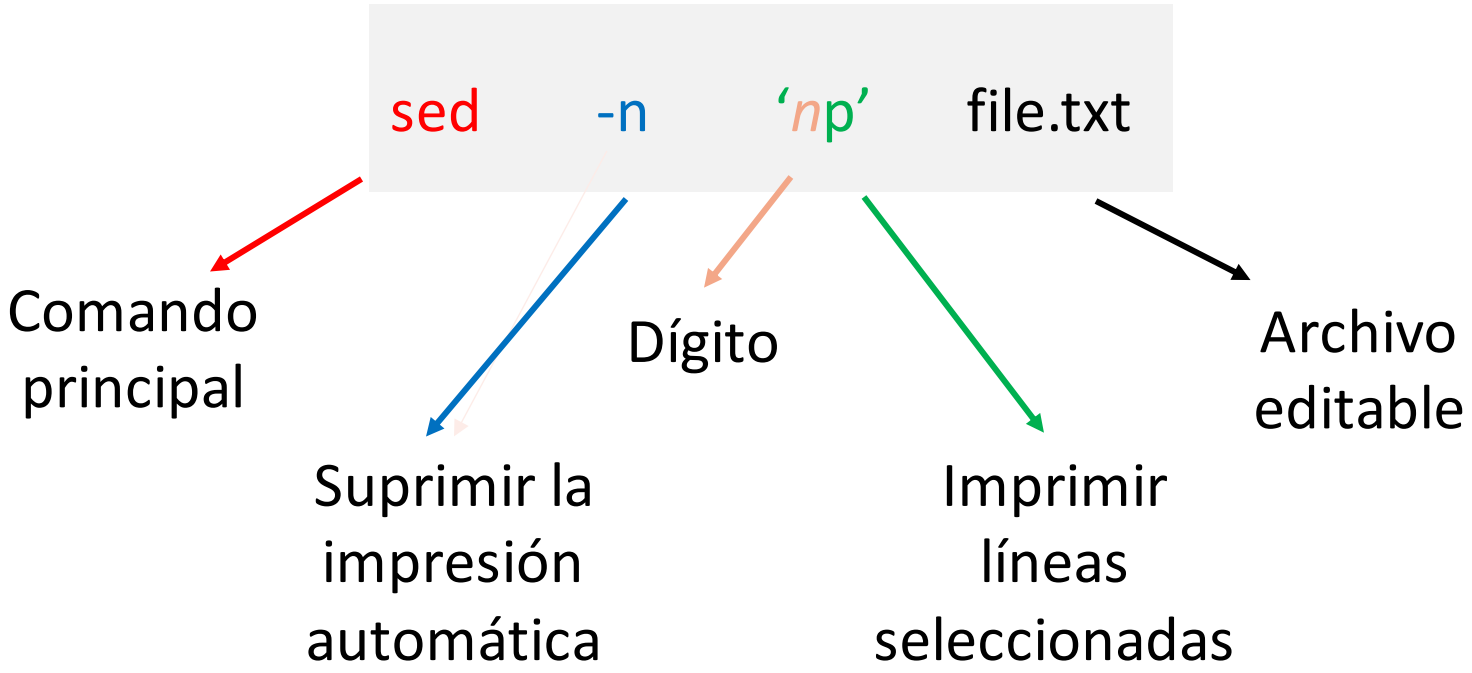
```
$ sed '1p' dias_de_la_semana.txt
Lunes
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

sed –n ‘1p ’ dias_de_la_semana.txt



```
$ sed –n '1p' dias_de_la_semana.txt
Lunes
```

Impresión de rangos de líneas sed



Impresión de rangos de líneas sed

- sed -n ‘1p’ diasSemana.txt # imprime la primera línea
- sed -n ‘1,5p’ diasSemana.txt # imprime las primeras 5 líneas
- sed -n ‘1,+4p’ diasSemana.txt # imprime las primeras 5 líneas
- sed -n ‘1~2p’ diasSemana.txt # imprime cada 2 líneas
- sed -n -e ‘2p’ -e ‘4p’ diasSemana.txt # imprime líneas específicas

Comando sed : ‘d’ *(delete)* -> Eliminación específica de líneas de un archivo

```
sed -n ‘1d’      diasSemana.txt   # elimina la primera línea

sed -n ‘1,5d’    diasSemana.txt   # elimina las primeras 5 líneas

sed -n ‘1,+4d’   diasSemana.txt   # elimina las primeras 5 líneas

sed -n ‘1~2d’    diasSemana.txt   # elimina cada 2 líneas
```

Comando sed : ‘d’ *(delete)* -> Eliminación específica de líneas de un archivo

```
sed  ‘1d’      diasSemana.txt   # elimina la primera línea

sed  ‘1,5d’    diasSemana.txt   # elimina las primeras 5 líneas

sed  ‘1,+4d’   diasSemana.txt   # elimina las primeras 5 líneas

sed  ‘1~2d’    diasSemana.txt   # elimina cada 2 líneas
```

Para observar el resultado se debe eliminar la opción **-n**

Comando sed : ‘i’ *(insert)* -> Insertar líneas específicas de un archivo

```
$ cat dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

```
$ sed  '1i Días de la semana' dias_de_la_semana.txt
Días de la semana
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

Comando sed : insertar/modificar

```
$ sed '7i ++++' dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
++++
Domingo

$ sed '7a ++++' dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
++++
```

Comando sed : ‘c’ *(change)* -> Cambia líneas específicas de un archivo

```
$ cat dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

```
$ sed '7c Sabado_again' dias_de_la_semana.txt
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Sabado_again
```

```
$ sed -e '7c Sabado_again' -e '1c Domingo' dias_de_la_semana.txt
Domingo
Martes
Miércoles
Jueves
Viernes
Sábado
Sabado_again
```

Comando sed

Busca patrones de texto usando **patrones** y **expresiones regulares** .

sed '/texto_a_buscar/p' fichero > fichero_nuevo

- ' ' : comillas simples donde introducimos el texto a buscar y el texto a reemplazar.
- / / : delimitadores
- p: imprimir las líneas coincidentes.
- <fichero>: Es nombre del fichero en que se buscan las partes de texto.
- >fichero_nuevo : Es un nuevo fichero que se generará con el texto localizado.

```
$ sed -n '/Lunes/p' dias_de_la_semana.txt
Lunes
```

```
$ sed -n '/^[MN]/p' dias_de_la_semana.txt
Martes
Miércoles
```

Comando sed : ‘s’ *(substitute)*

Función mas importante del Sed

sed permite la **sustitución de texto**

Busca patrones de texto y **reemplaza** el texto encontrado.

```
sed 's/texto_a_buscar/texto_a_reemplazar/' fichero_a_reemplazar > fichero_nuevo
```

- ' ' : comillas simples donde introducimos el texto a buscar y el texto a reemplazar.
- / / /: Dentro del primer delimitador incluimos la cadena de texto a buscar y dentro del segundo la cadena de texto que reemplaza a la que estamos buscando.
- s indica que queremos buscar y reemplazar.
- <fichero_a_reemplazar: Es nombre del fichero en que se buscan las partes de texto a modificar.
- >fichero_nuevo : Es un nuevo fichero que se generará con el texto reemplazado.

Sustitución básica de texto ‘s’ *(substitute)*

```
$ sed 's/Lunes/Inicio de semana/' dias_de_la_semana.txt
Inicio de semana
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo

$ sed -n 's/Lunes/Inicio de semana/' dias_de_la_semana.txt

$ sed -n 's/Lunes/Inicio de semana/p' dias_de_la_semana.txt

Inicio de semana
```

Comando sed : ‘s’ *(substitute)*

1. ***sed*** reemplaza patrones, no palabras.
2. Por defecto, la orden **s opera en la primera coincidencia** de una línea y luego pasa a la siguiente línea.
3. Para que ***sed*** **reemplace cada instancia** de un patrón en lugar de solo la primera instancia en cada línea, debe añadir la **orden opcional g** al comando **s**

```
sed 's/on/forward/g’ song.txt
```

Comando sed : 's' *(substitute)*

```
$ cat song.txt
this is the song that never ends
yes, it goes on and on, my friend
some people started saying it
not knowing what it was
and they'll continue saying it forever
just because...
```

sed 's/on/ON/' song.txt

this is the s**ON**g that never ends
yes, it goes **ON** and on, my friend
some people started saying it
not knowing what it was
and they'll c**ON**tinue saying it forever
just because...

sed 's/on/ON/g' song.txt

this is the s**ON**g that never ends
yes, it goes **ON** and **ON**, my friend
some people started saying it
not knowing what it was
and they'll c**ON**tinue saying it forever
just because...

Comando sed -i

Cuando usamos **sed** es importante tener en cuenta que nuestro archivo **fuente no se ve afectado**

Las ediciones que se realizan con los comandos de sed **se envían a la salida estándar**

Si queremos **guardar nuestras ediciones**, podemos **redirigir** la salida estándar a un archivo usando el metacarácter > o usar **la opción –i** que alterará el archivo fuente.

```
sed -i '1~2d' diasSemana.txt    # elimina cada dos líneas y modifica diasSemana.txt
```

Creación de un archivo de respaldo (*history*)

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ sed -i '1~2d' dias_de_la_semana.txt
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ cat dias_de_la_semana.txt
Martes
Jueves
Sábado
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ vim dias_de_la_semana.txt
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ sed -i.back '1~2d' dias_de_la_semana.txt
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ ls
comandos     dates.txt              Downloads          lista_secundaria  Public        script.sh          Sesion5              words.txt
comandos.txt  Desktop                expression.txt     Music             references2.txt  sequences.bis.sort.txt  Templates
core.31938   dias_de_la_semana.txt     gencode_annotation_42  numeros.txt     references.txt   sequences.bis.txt     uniq.txt
core.4554    dias_de_la_semana.txt.back  human_genome_hg38     num_genes.txt    samples.txt      sequences.sort.txt    variables_especiales.sh
data.txt     Documents              lista_principal.txt  Pictures          Scripts          sequences.txt         Videos
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ cat dias_de_la_semana.txt
Martes
Jueves
Sábado
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ cat dias_de_la_semana.txt.back
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
```

Ejercicio final: Comando sed

```
> $ cat data.txt
seq1    chr1    19
seq2    chr1    33
seq3    chr12    36

seq4    chr13    100
seq5    chr13    150

seq6    chr3    200
```

1. Reemplazar los tabuladores por comas

sed 's/\t/,/g' data.txt

2. Imprimir la línea 3 y 5

sed -n -e '3p' -e '5p' data.txt

sed -n ‘3p;5p' data.txt

3. Reemplazar todos los números por A

sed 's/[0-9]/A/g' data.txt

4. Eliminar las líneas vacías.

sed -i '/^$/d' data.txt

Si eliminas el 200 con este comando: sed ‘/200/d’ data.txt, no estas eliminando solo el numero 200, si no toda la línea, porque SED TRABAJA CON LÍNEAS

Convirtiendo un archivo FASTQ a un archivo FASTA

```
@SRR1984406.1 1 length=135
GACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTTCCAGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTG
+
CCCFFFFFHHHHHJJJJIJGHHHJJJJJJJJJIJIJJJJJJJJJJJIIJJJJJCHIJJJJJJHHHFFFFFFEEEFEEEDCC>CDDDDDDDCCCD:<@D@CDDDDDCC

@SRR1984406.2 2 length=134
TTTGGGAATTTCCTGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCA
+
CCCFFFFFHHHHHJJIIJJJJJJJJJJJJJJJJJJJJJJJJJIIIJJJJJJJJJJJJJJIJJJJIJJJHHHHFFFFEEEEEEDDDDBDDDDDDDDDDDDDDDCCDDDDDDD
```

```
>SRR1984406.1 1 length=135
GACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTTCCAGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTG

>SRR1984406.2 2 length=134
TTTGGGAATTTCCTGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCA
```

Aspectos a tratar

1. Comandos para el procesamiento de datos:
   - **grep**
   - **sed**
   - **awk**

awk

awk es un lenguaje de programación y un **procesador de texto** que se puede utilizar para manipular datos de texto

Excelente para analizar y manipular **archivos de texto**

Opera línea por línea y recorre todo el archivo.

Sintaxis de awk

```
awk '/patrón_búsqueda/{acción_a_realizar_coincidencia;otra_acción}' archivo
```

- La **búsqueda** se especifica entre **barras /**
- La **acción** a realizar se indica entre **llaves** separando cada acción con **punto y coma ;**

```
awk '/patron_busqueda/{acción_a_realizar_coincidencia;otra_acción}' archivo
```

```
awk '/patrón_búsqueda/{acción_a_realizar_coincidencia;otra_acción}' archivo
```

Sintaxis de awk

Omitiendo el patrón de búsqueda

```
$ awk '{print}' song.txt
this is the song that never ends
yes, it goes on and on, my friend
some people started saying it
not knowing what it was
and they'll continue saying it forever
just because...
```

awk ~~'/patron_busqueda/{~~acción_a_realizar_coincidencia;otra_acción~~}'~~ archivo

Omitiendo la acción a realizar

```
$ awk '/people/' song.txt
some people started saying it
```

awk '/patrón_búsqueda/{~~acción_a_realizar_coincidencia;otra_acción~~}' archivo

Sintaxis de awk

```
$ awk '{print}' song.txt
this is the song that never ends
yes, it goes on and on, my friend
some people started saying it
not knowing what it was
and they'll continue saying it forever
just because...
```

awk '/s$/{print $1}' archivo

Líneas que finalicen con el carácter s

Impresión específicamente el primer campo

Sintaxis de awk

```
$ awk '{print}' song.txt
this is the song that never ends
yes, it goes on and on, my friend
some people started saying it
not knowing what it was
and they'll continue saying it forever
just because...
```

```
$ awk '/s$/{print $1}' song.txt
this
not

$ awk '/s$/{print $2}' song.txt
is
knowing

$ awk '/s$/{print $0}' song.txt
this is the song that never ends
not knowing what it was
```

Es como si no pusieras nada, solo print e imprime toda la línea

Sintaxis de awk (opción -F = Field separator)

```
$ cat word_guion.txt
amarillo-3
azul-7
verde-4
negro-6
```

```
$ awk '/^a/{print $1}' word_guion.txt
amarillo-3
azul-7
```

EL DELIMITADOR QUE ENTIENDE ES POR ESPACIOS, NO GUION, POR TANTO TE LO VA A IMPRIMIR COMO SI FUERA LA MISMA PALABRA

```
$ awk -F '-' '/^a/{print $1}' word_guion.txt
amarillo
azul
$ awk -F '-' '/^a/{print $2}' word_guion.txt
3
7
```

y aqui le defines el separador

Sintaxis de awk (opción –F / VARIABLES INTERNAS)

OPCIONES

```
$ awk -F '-' '/^a/{print $1}' word_guion.txt
amarillo
azul
$ awk -F '-' '/^a/{print $2}' word_guion.txt
3
7
```

VARIABLES INTERNAS PREDEFINIDAS (FS => FIELD SEPARATOR)

```
$ awk -v FS='-' '/^a/{print $2}' word_guion.txt
3
7
```

Sin espacios en blanco

Variables internas predefinidas en AWK

BINMODE: Especifica el modo binario de I/O en sistemas no Unix: 1,2,3
CONVFMT: Controla la conversión de strings a números, pasa a sprintf
FIELDWIDTHS: Lista de columnas separadas cuando no es delimitada la entrada
FPAT: Regex que reconoce los campos para separarlos
FS: Separador de los campos de entrada
IGNORECASE: Si es no cero o no nulo, las comparaciones son case-indep
LINT: Cuando esta variable es verdadera, activa la opción de lint
OFMT: Controla la conversión de números a strings
OFS: Controla el separador de la salida
ORS: Separador de final de línea de cada registro
PREC: Precisión en las operaciones de punto flotante
ROUNDMODE: Modo de redondeo de números
RS: Separador de registros en la entrada
SUBSEP: Separador de subscripts en arrays multidimensionales
TEXTDOMAIN: Se usa para la internacionalización de programas de AWK
ARGC: Número de argumentos de línea de comandos
ARGV: Lista de argumentos de línea de comandos
ARGIND: índice de ARGV del archivo que está siendo procesado
ENVIRON: Array con los valores de las variables de ambiente
ERRNO: Guarda el valor del error durante un getline o close
FILENAME: Nombre del archivo que está siendo procesado
FNR: Número de registro que está siendo procesado
NF: Número de campos en el registro que está siendo procesado
FUNCTAB: Array con valores de funciones utilizadas
NR: Número de registros que awk ha procesado desde el inicio
PROCINFO: Array con información sobre el programa de awk
RLENGTH: Longitud de la cadena encontrada por un regex
RSTART: Indice del substring donde comienza un match
RT: Texto que se encuentra usando RS como separador
SYMTAB: Array con variables globales y arrays en el programa

AWK como lenguaje de programación: Variables internas

```
$ awk '{print $0}' references2.txt
seq1 ref2 K562 10
seq2 ref800 K562 20
seq2 ref800 GM1787 50
seq3 ref201 GM1787 100

$ awk '{print NF}' references2.txt  -> (field number/columnas)
4
4
4
4
```

siempre que esté separado por espacios

```
$ awk '{print NR}' references2.txt  -> (register number/filas)
1
2
3
4
```

AWK como lenguaje de programación: Criterios

```
$ cat references2.txt
seq1 ref2 K562 10 20
seq2 ref800 K562,20
seq2 ref800 GM1787 50
seq3 ref201 GM1787 10

$ awk '{print NF}' references2.txt
5
3
4
4
```

CRITERIOS

```
$ awk 'NF > 4 {print $0}' references2.txt
seq1 ref2 K562 10 20
```

<	Menor que
>	Mayor que
<=	Menor o igual que

>=	Mayor o igual que
==	Igual a
!=	desigual a

AWK como lenguaje de programación

```
$ ls -lh | head
total 415M
drwxr-xr-x  4 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users 4.0K Oct 23 09:21 comandos
-rw-r--r--  1 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users  26K Oct 24 13:52 comandos.txt
-rw-------  1 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users 318M Jul 13 15:49 core.31938
-rw-------  1 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users 204M Aug 17 16:56 core.4554
-rw-r--r--  1 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users   72 Oct 26 10:26 data.txt
-rw-r--r--  1 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users  189 Oct 26 18:13 dates.txt
drwxr-xr-x  2 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users 4.0K Jul 13 15:48 Desktop
-rw-r--r--  1 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users   56 Nov  3 11:40 dias_de_la_semana.txt
drwxr-xr-x  6 UNIVERSIDADVIU\paula.soler  UNIVERSIDADVIU\domain  users 4.0K Oct 20 08:19 Documents

$ ls -lh | awk '{print $6}' | head

4.0K
26K
318M
204M
72
189
4.0K
56
4.0K
```

AWK como lenguaje de programación: Elementos de control

```
$ awk '{print $0}' references2.txt
seq1 ref2 K562 10
seq2 ref800 K562 20
seq2 ref800 GM1787 50
seq3 ref201 GM1787 100


$ awk '{if ($4 > 20) print $0}' references2.txt
seq2 ref800 GM1787 50
seq3 ref201 GM1787 100
```

"Si algun valor de la columna 4 es mayor que 20, imprime toda la fila"

- if (expr) statement
- if (expr) statement else statement
- while (expr) statement
- do statement while (expr)
- for (opt_expr ; opt_expr ; opt_expr) statement
- for (var in array) statement

Lenguaje de programación: Scripts con AWK

También hay bloques **BEGIN** y **END** opcionales que pueden contener comandos para ejecutar antes y después del procesamiento del archivo, respectivamente.

- La sección **BEGIN** se ejecuta antes del tratamiento del primer registro de datos. Se usa esencialmente para iniciar el contexto de ejecución.
- Puede haber varias secciones intermedias que se ejecutarán sobre cada registro.
- La sección **END** se ejecuta después del tratamiento del último registro de datos. Se usa para explotar los resultados obtenidos del tratamiento de datos.

Lenguaje de programación: Scripts con AWK

```
$ awk '{print $0}' references2.txt
seq1 ref2 K562 10
seq2 ref800 K562 20
seq2 ref800 GM1787 50
seq3 ref201 GM1787 100
```



```
> awk 'BEGIN{suma=0}{suma += $4}END{print suma}' references2.txt
180
```

Ejercicio final -> Comando awk

```
> $ cat data.txt
seq1 chr1 19
seq3 chr1 34
seq2 chr2 182

seq1 chr10 55
seq2 chr11 33

seq4 chr3 22
```

1. Imprime las líneas que no contengan el 33 en el campo 3.

awk '{if($3!=33) print $0}' data.txt

2. Imprime las líneas que contengan el 182 en el campo 3.

awk '{if($3==182) print $0}' data.txt

3. Imprime las líneas que contengan un valor mayor o igual a 50 en el campo 3.

awk '{if($3>=50) print $0}' data.txt

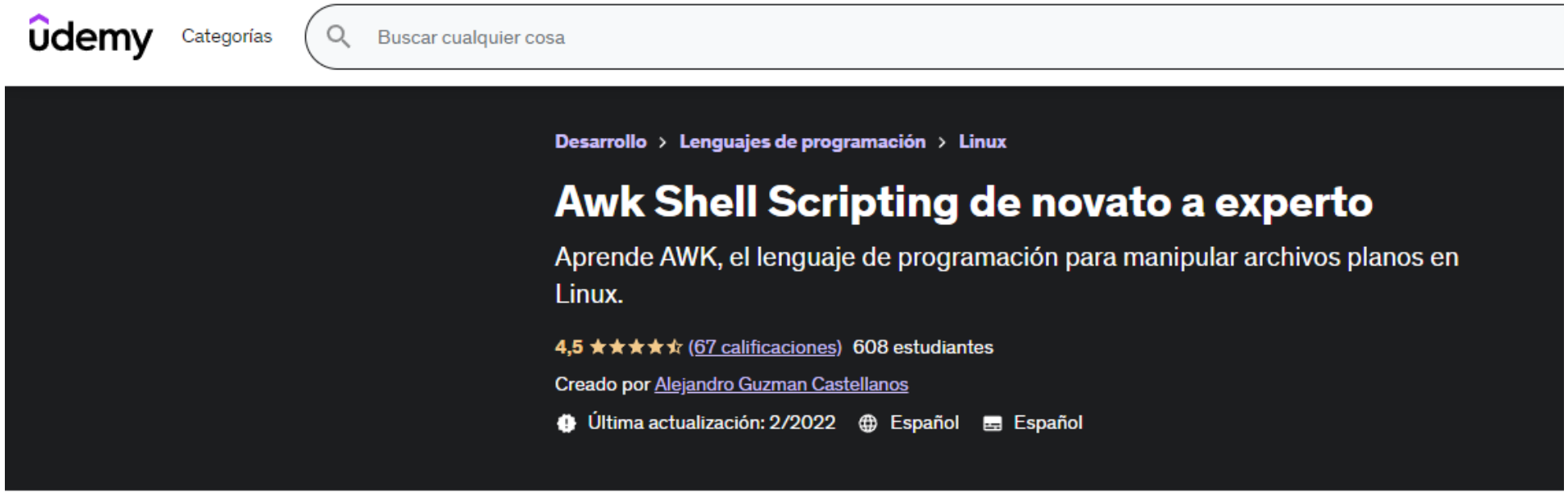
Ejercicio final -> Comando awk

```
> $ cat data.txt
seq1 chr1 19
seq3 chr1 34
seq2 chr2 182

seq1 chr10 55
seq2 chr11 33

seq4 chr3 22
```

4. Calcula el promedio del campo 3

```
sed '/^$/d' data.txt | awk 'BEGIN{valor=0}{valor += $3}END{print valor/NR}'
```

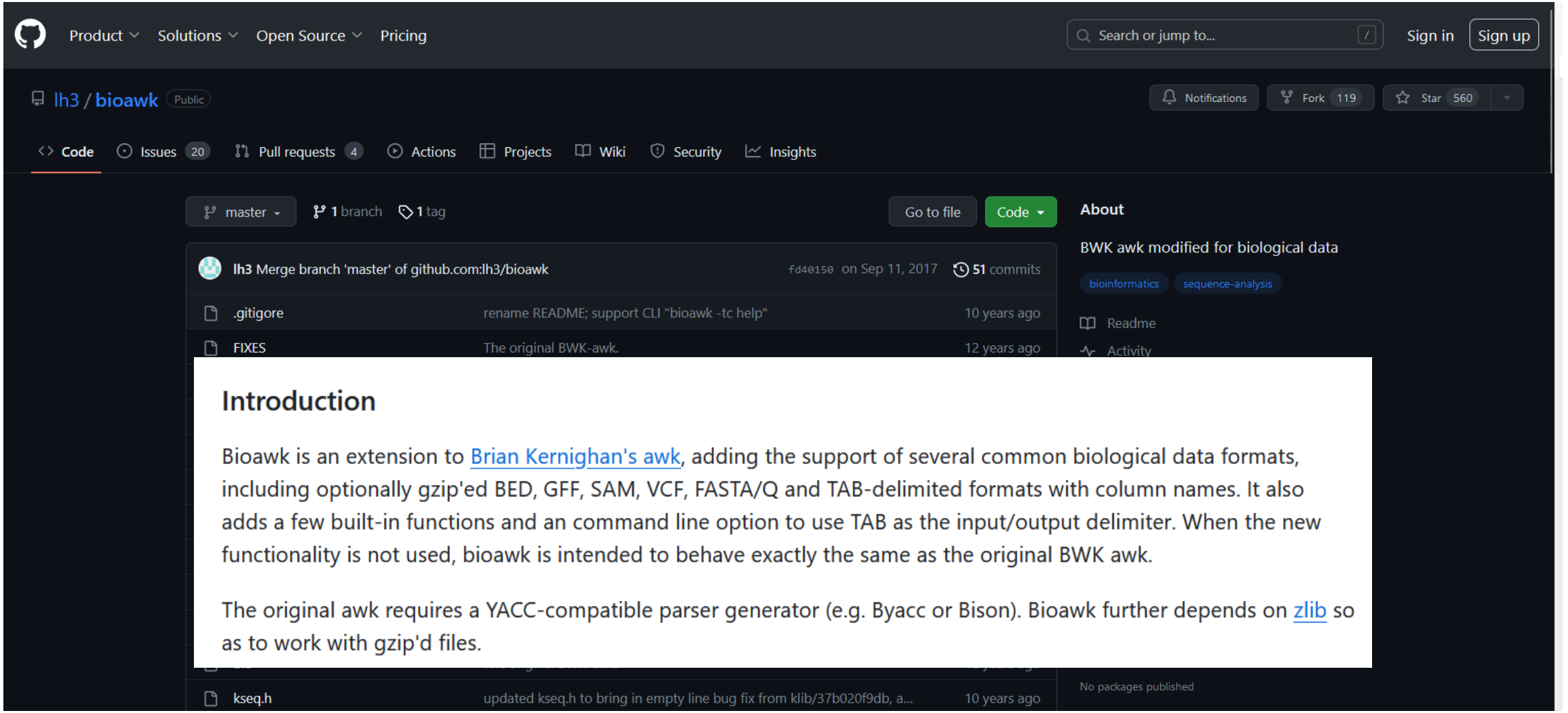
Cursos de AWK Shell Scripting



Lo que aprenderás

- ✓ Aprenderas hacer operaciones rápidas sobre archivos estructurados.
- ✓ Aprenderás a utilizar un lenguaje de programación orientado a búsqueda de patrones y operaciones especificas sobre el resultado.
- ✓ Dominaras un lenguaje sencillo pero poderoso para realizar diversas tareas de básicas a completas en la administración de Sistemas.

BIOAWK



https://github.com/lh3/bioawk

Universidad
Internacional
de Valencia

universidadviu.com

De:
Planeta Formación y Universidades