Programación con Shell Scripting: Sesión 12

Máster Universitario en Bioinformática

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

De:
Planeta Formación y Universidades

Prueba aplicativa 2

Prueba aplicativa 2

Archivos adjuntos: Actividad 2.docx (17,94 KB)
Actividad 2.pdf (361,135 KB)
SRR1984406_1.fastq (2,582 MB)

Actividad 2.- Manipulación y formateo de archivos: Formatos FASTQ y FASTA

El propósito de esta actividad es desarrollar un flujo de trabajo bioinformático integral, conocido como pipeline, para procesar datos biológicos. La intención es que el estudiante adquiera destrezas para interactuar con el sistema operativo mediante la línea de comandos y pueda desarrollar scripts en Shell para abordar diferentes desafíos bioinformáticos, centrándose en dos formatos de texto específicos: FASTQ y FASTA.

Instrucciones de entrega

- La entrega se realizará a través del Campus VIU en un archivo único en formato PDF utilizando este documento como plantilla.
- Incluya el código empleado, capturas de pantalla con su usuario (agregando el *promp* completo) y resolución máxima.
- Proporcione explicaciones claras y concisas de los comandos utilizados. Si los comandos empleados no se explican brevemente, el valor de la pregunta será penalizado a la mitad.
- Reportar solo una opción/forma para resolver las distintas preguntas planteadas.

Universidad Internacional de Valencia

Aspectos a tratar

1. Sentencias de control de flujo:
   - **Bucles o *loops***
   - *for*
   - *while*

2. Sentencias de control de flujo:
   - **Condicionales**
   - *if/else/elif*

EL BUCLE WHILE

El bucle WHILE: sintaxis

Se usa para ejecutar un conjunto de comandos dado un número desconocido de veces, siempre que la condición especificada sea verdadera.

SINTAXIS BÁSICA

```
while [ CONDITION(s) ]
do
        command-1
        command-2
        command-n
done
```

Espacios en blanco

El bucle WHILE: ejemplo

```
#!/bin/bash

number=10
while [ $number -gt 5 ]

do
    echo $number
    number=$(($number-1))

done
```

Se recomienda usar paréntesis siempre con WHILE

```
$ bash test.sh
10
9
8
7
6
```

El bucle WHILE: ejemplo

```
#!/bin/bash

number=10
while (( $number > 5 ))

do
    echo $number
    number=$(($number-1))

done
```



```
$ bash test.sh
10
9
8
7
6
```

Operadores de comparación para valores numéricos [ ]

Operador	Significado
-lt	Menos que (<)
-gt	Mayor que (>)
-le	Menor o igual que (≤)
-ge	Mayor o igual que (≥)
-eq	Igual (=)
-ne	No igual (≠)

Forever while loop

```
#!/bin/bash


while [ true ]
do
  echo "This is an infinite while loop. Press CTRL + C to exit out of the loop."
  sleep 0.5
done
```

BOOLEANOS

cuando se habla de valores boleanos solo tiene dos opciones: o true o false

Sintaxis de los bucles

```
for i in 1 2 3 4 5 .. N


do
    command-1
    command-2
    command-n


done
```

```
while [ CONDITION ]


do
        command-1
        command-2
        command-n


done
```

Leyendo archivos con bucles

```
#!/bin/bash

file="casa armario compra codo euro"

for var in $file; do
    echo "$var"
done
```

```
$ bash script.sh
casa
armario
compra
codo
euro
```

El separador de campo (IFS)

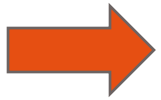
De forma predeterminada, los siguientes caracteres se tratan como campos.

- Espacio
- Tabulación
- Nueva línea

IFS (Internal Field Separator)

- Variable de entorno que define los caracteres que el shell considera como separadores de campos al leer una línea de texto.

- Por defecto, su valor es un espacio, un tabulador y un salto de línea



```
# Guardar el valor original de IFS
OLD_IFS=$IFS
# Cambiar el valor de IFS temporalmente
IFS=:
# Restaurar el valor original de IFS
 IFS=$OLD_IFS
```

Leyendo archivos línea a línea con bucles FOR

1

```
$ cat file.txt
Esta es la primera secuencia
Esta es la segunda secuencia


$ cat script.sh
#!/usr/bin/bash
for line in $(cat file.txt)
do
        echo ${line}
done
```

2

```
$ cat script2.sh
#!/usr/bin/bash
oldIFS=$IFS
IFS=$'\n'
for line in $(cat file.txt)
do
        echo ${line}
done
IFS=$oldIFS
```

para poder eliminar el salto de línea hay que indicárselo con $'\n'

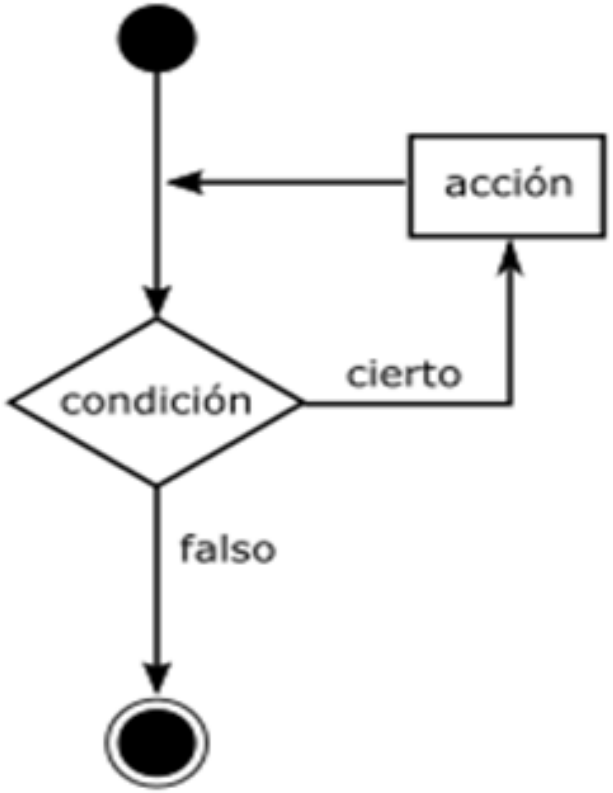
la mejor solución es utilizando el comando WHILE

Leyendo archivos línea a línea con bucles WHILE

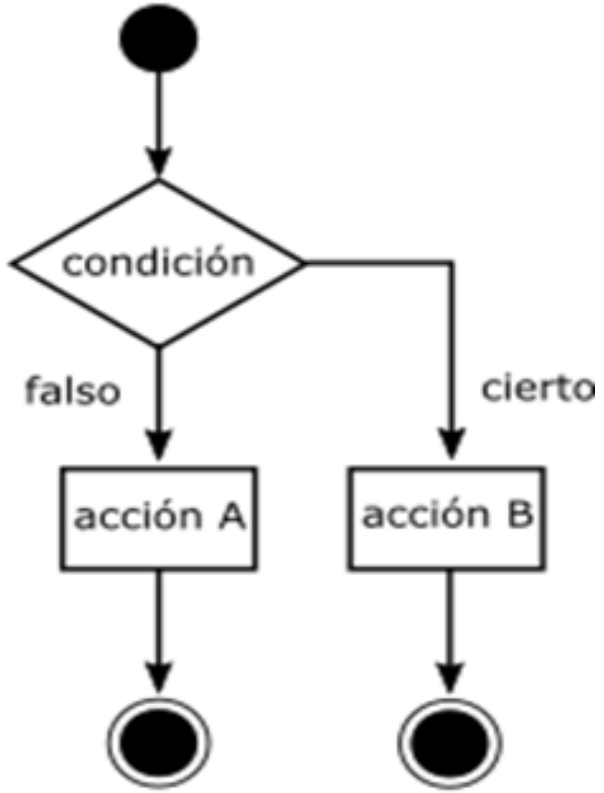
```
$ cat file.txt
Esta es la primera secuencia
Esta es la segunda secuencia

$ cat script3.sh
#!/usr/bin/bash

cat file.txt | while read line
do
   echo $line
done
##########################################################################################

while read line
do
   echo $line
done < file.txt
```

Control de flujo

Ejecución
línea a línea



Bucle (*for*)

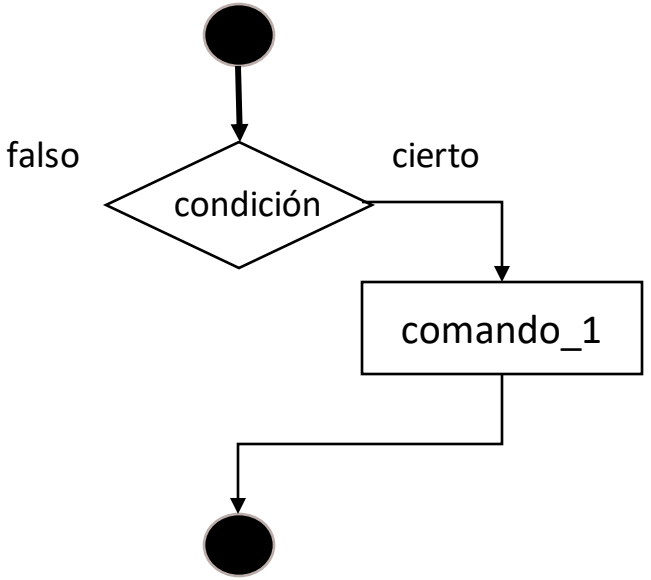


Condicional (*if*)

Condición simple (if…then)

Se usa para ejecutar un conjunto de comandos solamente si se cumplen una condición determinada

```
if [ condition-statement ]

then
    command1 # si se cumple la condición
    command2
    commandN

fi
```



Condición simple

```
#!/usr/bin/bash
echo "Enter your marks out of 100: "
read marks
if [ $marks -gt 100 ]
then
        echo "You have entered incorrect marks: $marks"
fi
```

Relación entre el if y test en Shell

```
#!/usr/bin/bash
echo "Enter your marks out of 100: "
read marks
if [ $marks -gt 100 ]
then
        echo "You have entered incorrect marks: $marks"
fi
```

Toma decisiones basadas en el resultado de la evaluación de esas expresiones.

Evalúa expresiones condicionales

[ ] = test

los corchetes son sinónimo de test

Comando test

Comando que devuelve un valor indicando si su función/comparación ha tenido éxito o no

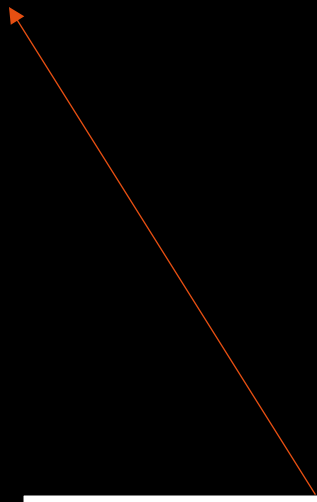
Sintaxis

```
test EXPRESSION

[ EXPRESSION ]
```

Comando test

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ test 3 -eq 3
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ [ 3 -eq 3 ]
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ [3 -eq 3] #Dejar siempre espacios en blanco
```

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ test 3 -eq 3 ; echo “$?”
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ test 3 -ne 3 ; echo "$?"
1 # False
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ test 3 -gt 1 ; echo "$?"
0 # True
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ test 3 -lt 1 ; echo "$?"
1 # False
```



Variables especiales
Código de salida

Expresiones comparativas numéricas

```
#!/bin/bash

[ $1 -eq $2 ] && echo "$1=$2"
[ $1 -ne $2 ] && echo "$1!=$2"
[ $1 -lt $2 ] && echo "$1<$2"
[ $1 -gt $2 ] && echo "$1>$2"
```

compare_number.sh

$1 - $9 (**Argumentos** de un script)

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ ./compara_numeros.sh 3 1
3!=1
3>1
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ ./compara_numeros.sh 1 5
1!=5
1<5
```

Expresiones comparativas numéricas

```
#!/bin/bash

[ $1 -eq $2 ] && echo "$1=$2"
[ $1 -ne $2 ] && echo "$1!=$2"
[ $1 -lt $2 ] && echo "$1<$2"
[ $1 -gt $2 ] && echo "$1>$2"
```

compare_number.sh

Ejecución condicional

- **Operador lógico Y (&&):** El comando anterior ha de finalizar exitosamente

$ comando1 && comando2

- **Operador lógico O (||)**: El comando anterior no ha de finalizar exitosamente

$ comando1 || comando2

Expresiones condicionales

- **Expresiones comparativas numéricas.**
- **Expresiones comparativas de caracteres.**
- **Expresiones con ficheros.**

Operadores de comparación para texto

Operador	Significado
=	Las dos cadenas de texto son idénticas
!=	Las dos cadenas de texto no son idénticas
<	Es menor que
>	Es mayor que
-n	La cadena no está vacía
-z	La cadena está vacía

Operadores de comparación para texto

Operador	Significado
=	Las dos cadenas de texto son idénticas
!=	Las dos cadenas de texto no son idénticas
<	Es menor que
>	Es mayor que
-n	La cadena no está vacía
-z	La cadena está vacía

Comparación lexicográfica (alfabéticamente)

de dos palabras

Expresiones comparativas de cadenas

```
#!/bin/bash

cadena1=hola

cadena2=adios


[ $cadena1  =  $cadena2 ] && echo "$cadena1=$cadena2"

[ $cadena1  !=  $cadena2 ] && echo "$cadena1!=$cadena2"

[ -n $cadena1 ] && echo "El tamaño de $cadena1 no es 0"

[ $cadena1 > $cadena2 ] && echo "$cadena2 precede alfabéticamente a $cadena1"
```

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ ./compara_palabras.sh
hola!=adios
El tamaño de hola no es 0
Adios precede alfabéticamente a hola
```

Expresiones comparativas de cadenas

```
#!/bin/bash
cadena1="hola que tal"
cadena2="hola que tal"


[ "$cadena1"  =  "$cadena2" ] && echo "$cadena1=$cadena2"
[ $cadena1  =  $cadena2 ] && echo "$cadena1=$cadena2"
```

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ bash comillas.sh
hola que tal=hola que tal
comillas.sh: line 6: [: too many arguments
```

Expresiones de archivos

Operador	Significado
-e name	*name* existe
-f name	*name* es un archivo normal (no es directorio)
-s name	*name* tiene un tamaño mayor que cero
-d name	*name* es un directorio
-r name	*name* tiene permiso de lectura para el usuario que ejecuta el script
-w name	*name* tiene permiso de escritura para el usuario que ejecuta el script
-x name	*name* tiene permiso de ejecución para el usuario que ejecuta el script

Expresiones de archivos

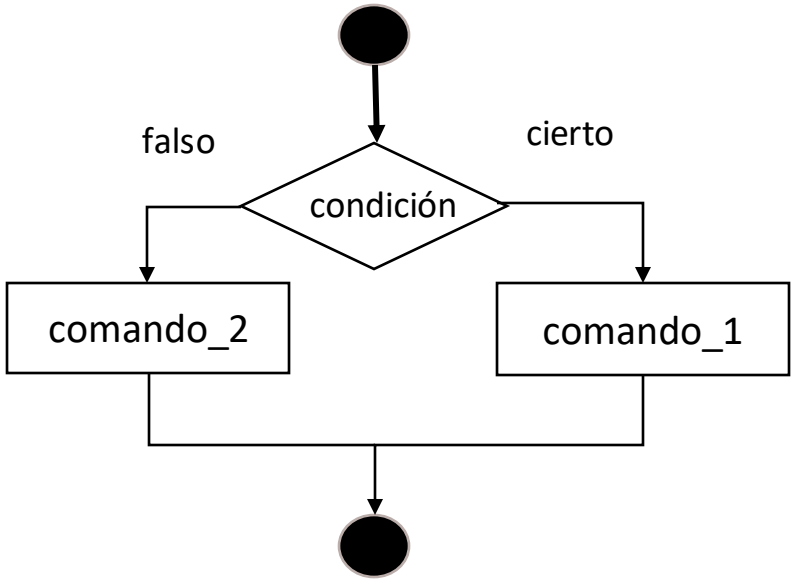
```
#!/usr/bin/bash

archivo=mi_archivo.txt

if test -e $archivo
then
    echo "El archivo $archivo existe"
else
    echo "El archivo $archivo no existe"
fi
```

=

```
#!/usr/bin/bash

archivo=mi_archivo.txt

if [ -e $archivo ]
then
    echo "El archivo $archivo existe"
else
    echo "El archivo $archivo no existe"
fi
```

Condiciones compuestas (if…then…else)

```
if [ condition-statement ]

then
    command1 # si se cumple la condición

else
    command2 # si no se cumple la condición

fi
```



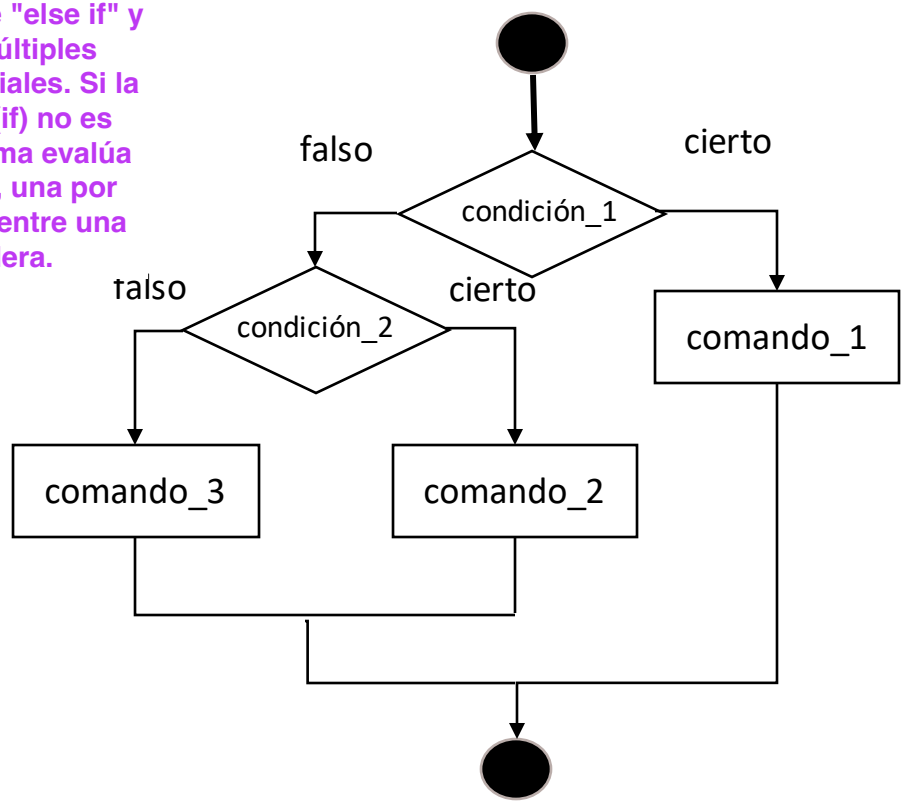
Declaración if con múltiples condiciones

```
#!/bin/bash
echo “Introduce un valor:"
read var
if [ $var -lt 10 ]
then
        echo "El valor es menor que 10"
else
        echo "El valor es mayor que 10"
fi
```

Condiciones anidadas (if, then y elif)

```
if [ condition-statement1 ]
then
    command1 # si se cumple la condición_1
elif [ condition-statement2 ] # else if
then
    command2 # si se cumple la condición_2
else
    command3 # si no se cumple la condición_2
fi
```

elif es una palabra clave en varios lenguajes de programación, como Python, que se utiliza para añadir condiciones adicionales a una estructura condicional if-else. Es una abreviatura de "else if" y permite evaluar múltiples condiciones secuenciales. Si la primera condición (if) no es verdadera, el programa evalúa las condiciones elif, una por una, hasta que encuentre una que sea verdadera.



Condiciones anidadas (if, then y elif)

```
#!/bin/bash
num1=$1          # la variable toma el primer valor que pasamos al script
num2=$2          # la variable toma el segundo valor que pasamos al script

if [ $num1 -gt $num2 ]
then
   echo $num1 es mayor que $num2
elif [ $num1 -lt $num2 ]
then
   echo $num1 es menor que $num2
else
   echo $num1 es igual a $num2
fi
```

comparar.sh

```
>./comparar.sh 10 5
10 es mayor que 5

>./comparar.sh 5 10
5 es menor que 10

>./comparar.sh 4 4
4 es igual a 4
```

Todo en juego *(for + if)*

```
#!/bin/bash
for file in $(ls)
do
  if [ -d $file ]                    # verifica si file es directorio
  then
    echo “directorio: $file”
  else
    if [ -x $file ]                  # verifica si se tiene permiso de ejecución
    then
      echo “archivo ejecutable: $file”
    else
      echo “archivo no ejecutable: $file”
    fi
  fi
done
```

Comandos basename/dirname

Estos comandos son muy básicos, y **sus funciones** son:

- **basename**: sirve para extraer el nombre del fichero de una ruta.
- **dirname**: sirve para extraer el nombre del directorio de una ruta.

```
$ basename /usr/bin/sort
sort
$ basename -a /usr/bin/sort  /etc/password
sort
password
$ basename -s .bed Downloads/*.bed
human_coordinates_1
human_coordinates_2
```

```
$ dirname /usr/bin/sort
/usr/bin
```

¿Qué realiza el siguiente Script en Shell?

```
#!/bin/bash

sed '1d' metadata.txt | awk '{print $1 "," $3}' > metadata.csv
mkdir archivos

for LINE in $(cat metadata.csv)
do

  ID=$(echo  $LINE | cut -d "," -f1)

  LINK=$(echo  $LINE | cut -d "," -f2)

  FILE=$(basename  $LINK)

  BASE=$(basename  -s .gz $LINK)

  echo $ID, $LINK, $FILE, $BASE




done
```

MCL1-DL,https://zenodo.org/record/4249555/files/SRR1552455.fastq.gz
MCL1-DK,https://zenodo.org/record/4249555/files/SRR1552454.fastq.gz
MCL1-DJ,https://zenodo.org/record/4249555/files/SRR1552453.fastq.gz
MCL1-DI,https://zenodo.org/record/4249555/files/SRR1552452.fastq.gz
MCL1-DH,https://zenodo.org/record/4249555/files/SRR1552451.fastq.gz
MCL1-DG,https://zenodo.org/record/4249555/files/SRR1552450.fastq.gz
MCL1-LF,https://zenodo.org/record/4249555/files/SRR1552449.fastq.gz
MCL1-LE,https://zenodo.org/record/4249555/files/SRR1552448.fastq.gz
MCL1-LD,https://zenodo.org/record/4249555/files/SRR1552447.fastq.gz
MCL1-LC,https://zenodo.org/record/4249555/files/SRR1552446.fastq.gz
MCL1-LB,https://zenodo.org/record/4249555/files/SRR1552445.fastq.gz
MCL1-LA,https://zenodo.org/record/4249555/files/SRR1552444.fastq.gz

metadata.csv

```
#!/bin/bash

sed '1d' metadata.txt | awk '{print $1 "," $3}' > metadata.csv
mkdir archivos

for LINE in $(cat metadata.csv)
do

  ID=$(echo  $LINE | cut -d "," -f1)

  LINK=$(echo  $LINE | cut -d "," -f2)

  FILE=$(basename  $LINK)

  BASE=$(basename  -s .gz $LINK)

  echo $ID, $LINK, $FILE, $BASE

done
```

MCL1-DL,https://zenodo.org/record/4249555/files/SRR1552455.fastq.gz

MCL1-DK,https://zenodo.org/record/4249555/files/SRR1552454.fastq.gz

MCL1-DJ,https://zenodo.org/record/4249555/files/SRR1552453.fastq.gz

MCL1-DI,https://zenodo.org/record/4249555/files/SRR1552452.fastq.gz

metadata.csv

MCL1-DL,https://zenodo.org/record/4249555/files/SRR1552455.fastq.gz, SRR1552455.fastq.gz, SRR1552455.fastq

Descarga Automática de Archivos FASTQ

```
#!/bin/bash

sed '1d' metadata.txt | awk '{print $1 "," $3}' > metadata.csv
mkdir archivos

for LINE in $(cat metadata.csv)
do

  ID=$(echo  $LINE | cut -d "," -f1)
  LINK=$(echo  $LINE | cut -d "," -f2)
  FILE=$(basename  $LINK)
  BASE=$(basename  -s .gz $LINK)

  echo $ID, $LINK, $FILE, $BASE
  if [ ! -f ./archivos/$ID_$BASE ]
  then
    echo " Descarga desde $LINK "
    wget $LINK
    echo " Descomprime $FILE "
    gunzip $FILE
    echo " Mueve al directorio"
    mv $BASE archivos/$ID_$BASE
  fi
done
```

MCL1-DL,https://zenodo.org/record/4249555/files/SRR1552455.fastq.gz, SRR1552455.fastq.gz, SRR1552455.fastq

Archivos/MCL1-DL_SRR1552455.fastq

Universidad
Internacional
de Valencia

universidadviu.com

De:
Planeta Formación y Universidades