Introducción a BASH

Francisco Alonso Sarria

Universidad de Murcia

Un problema

Tenemos 200 ficheros en formato TIF en un directorio y queremos transformarlos a JPG.

- ¿Cómo se hace en un programa visual?
- ¿Cómo se haría en un entorno basado en linea de comandos?

Un problema

Tenemos 200 ficheros en formato TIF en un directorio y queremos transformarlos a JPG.

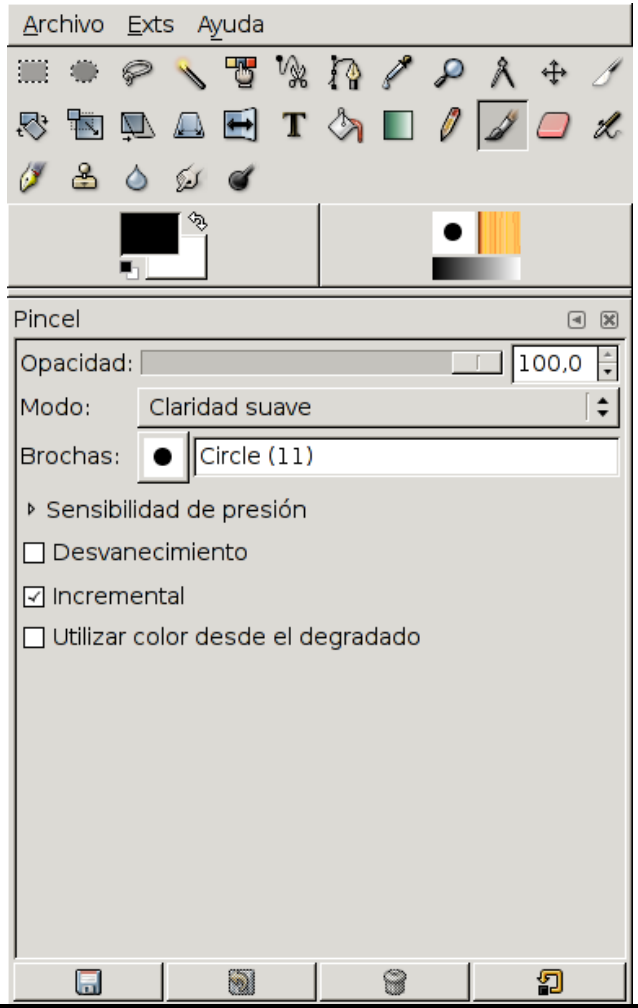
- ¿Cómo se hace en un programa visual?
- ¿Cómo se haría en un entorno basado en linea de comandos?

Un problema

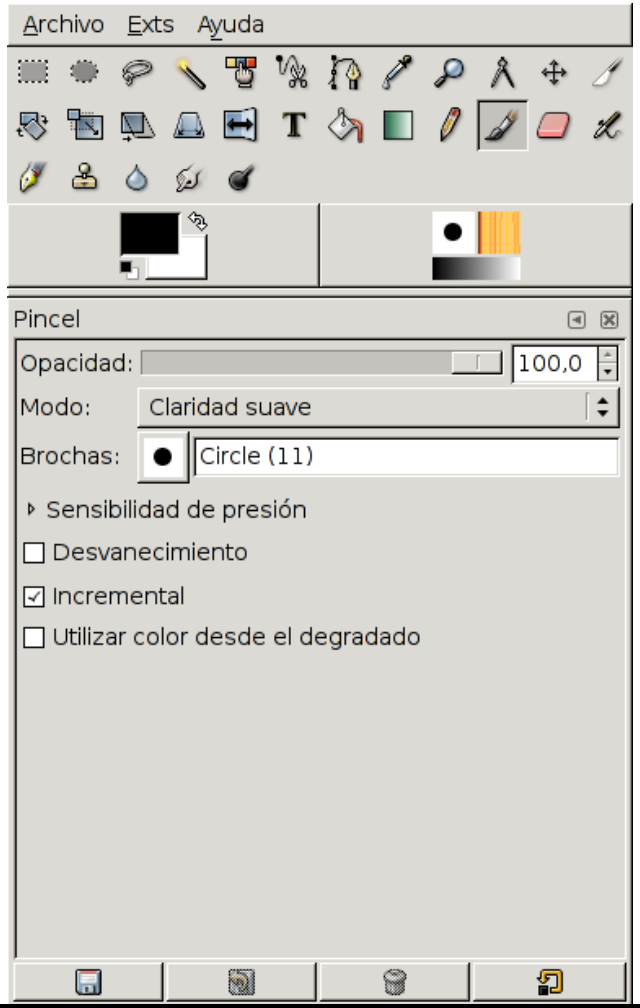
Tenemos 200 ficheros en formato TIF en un directorio y queremos transformarlos a JPG.

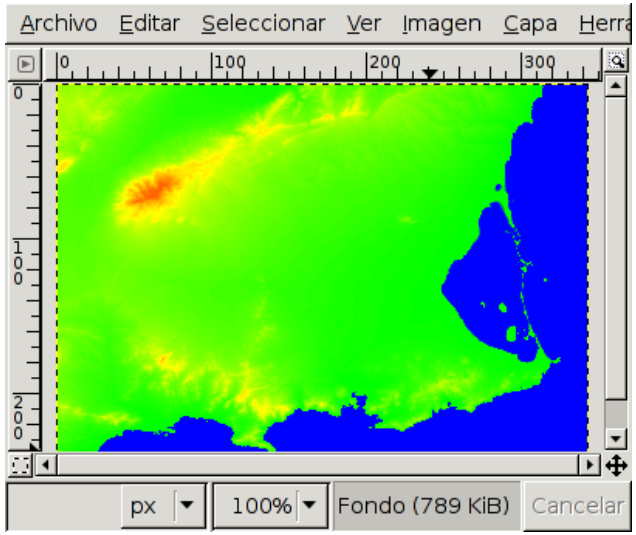
- ¿Cómo se hace en un programa visual?
- ¿Cómo se haría en un entorno basado en linea de comandos?

En un programa visual

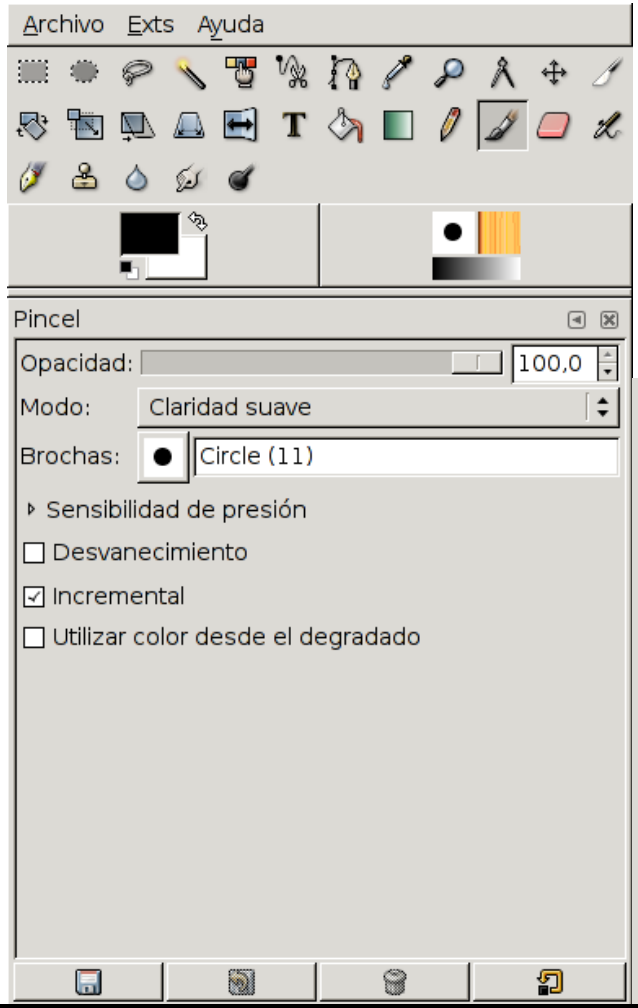


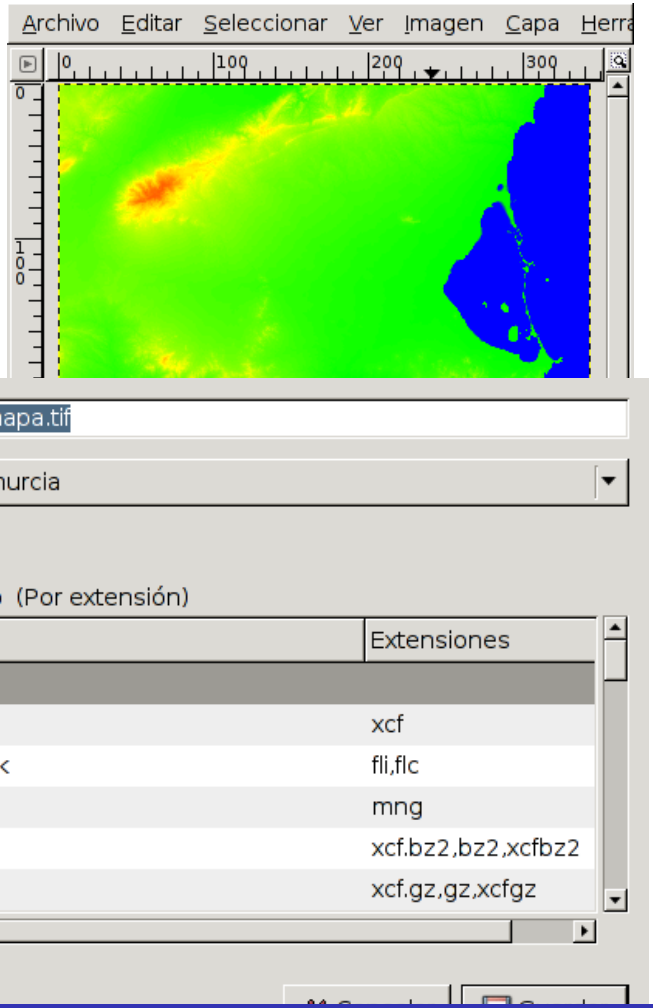
En un programa visual



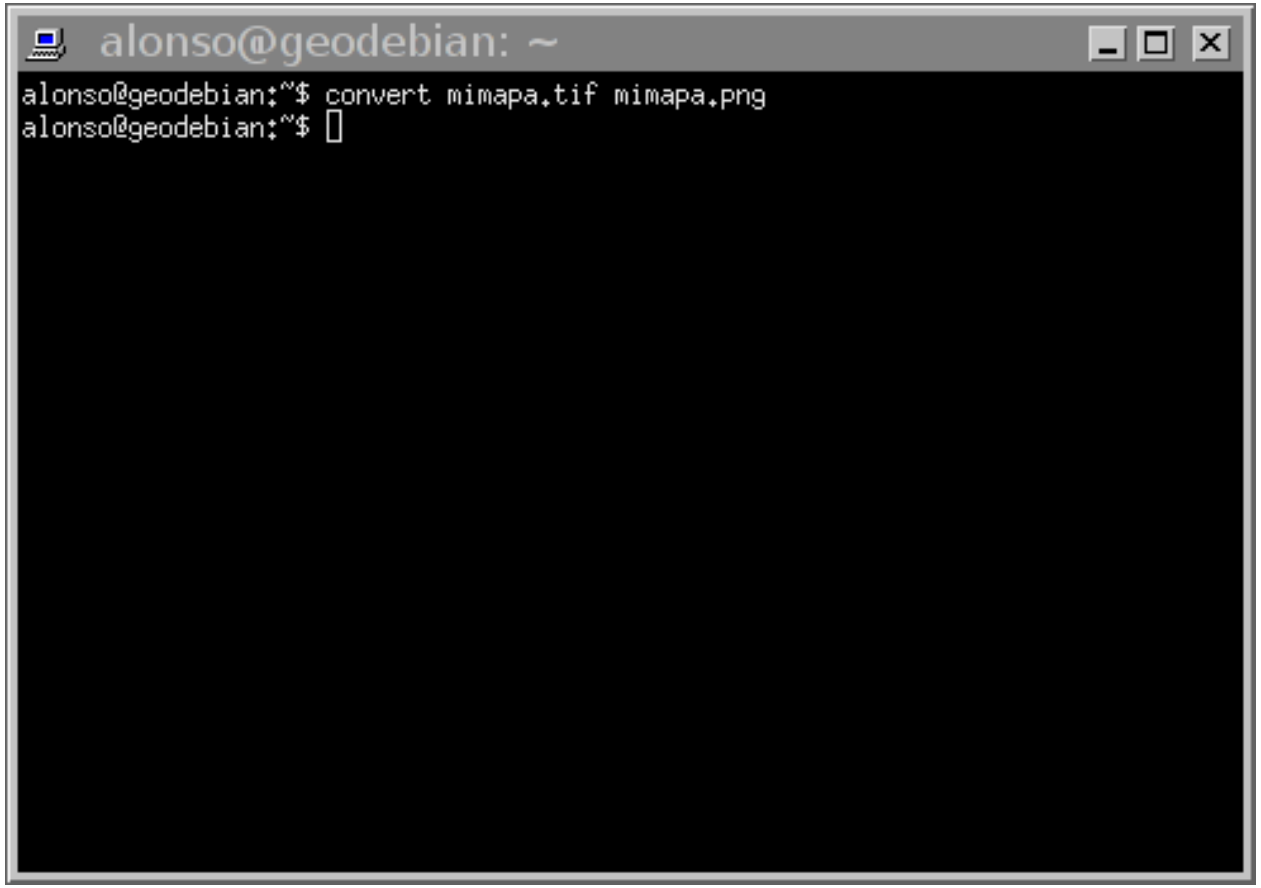


En un programa visual





En linea de comandos



Lista de tareas

```
convert fichero_1.tif fichero_1.jpg
convert fichero_2.tif fichero_2.jpg
convert fichero_3.tif fichero_3.jpg
   ..          ..          ..
convert fichero_N.tif fichero_N.jpg
```

Un script de BASH

Script:

```
for i in $(ls *.tif);do
      o=$(echo $i|sed 's/tif/jpg/');
      convert $i $o;
done
```

Un script de BASH

Script:

```
for i in $(ls *.tif);do
      o=$(echo $i|sed 's/tif/jpg/');
      convert $i $o;
done
```

Resultado:

Transformará todos los ficheros TIF del directorio de trabajo a formato JPG

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Operaciones con archivos

- Listado de ficheros: `ls`
- Editores de texto: `emacs`, `vi`
- Copiar, mover, renombrar y borrar archivos: `cp`, `mv`, `rm`, `mkdir`
- Visualización: `cat`, `more`, `less`, `head`, `tail`
- Partición del fichero en trozos: `split` (por filas), `cut` (por columnas)
- Concatenación: `cat` (por filas), `join` (por columnas)
- Consulta y sustitución: `sed`, `grep`
- Ordenación: `sort`
- Búsqueda de ayuda: `man`

Leer el contenido de un archivo

- `cat fichero`
  muestra el contenido de *fichero*.
- `more fichero`
  presenta el fichero página a página (pulsar una tecla para pasar página)
- `less fichero`
  similar a `more` pero con la posibilidad de ir arriba y abajo dentro del fichero
- `head -n fichero`
  presenta las `n` primeras lineas del fichero en pantalla
- `tail -n fichero`
  presenta las `n` últimas lineas del fichero en pantalla

Leer el contenido de un archivo

- `cat fichero`
  muestra el contenido de *fichero*.
- `more fichero`
  presenta el fichero página a página (pulsar una tecla para pasar página)
- `less fichero`
  similar a `more` pero con la posibilidad de ir arriba y abajo dentro del fichero
- `head -n fichero`
  presenta las `n` primeras lineas del fichero en pantalla
- `tail -n fichero`
  presenta las `n` últimas lineas del fichero en pantalla

Leer el contenido de un archivo

- `cat fichero`
  muestra el contenido de *fichero*.
- `more fichero`
  presenta el fichero página a página (pulsar una tecla para pasar página)
- `less fichero`
  similar a `more` pero con la posibilidad de ir arriba y abajo dentro del fichero
- `head -n fichero`
  presenta las `n` primeras lineas del fichero en pantalla
- `tail -n fichero`
  presenta las `n` últimas lineas del fichero en pantalla

Leer el contenido de un archivo

- `cat fichero`
  muestra el contenido de *fichero*.
- `more fichero`
  presenta el fichero página a página (pulsar una tecla para pasar página)
- `less fichero`
  similar a `more` pero con la posibilidad de ir arriba y abajo dentro del fichero
- `head -n fichero`
  presenta las `n` primeras lineas del fichero en pantalla
- `tail -n fichero`
  presenta las `n` últimas lineas del fichero en pantalla

Leer el contenido de un archivo

- `cat fichero`
  muestra el contenido de *fichero*.
- `more fichero`
  presenta el fichero página a página (pulsar una tecla para pasar página)
- `less fichero`
  similar a `more` pero con la posibilidad de ir arriba y abajo dentro del fichero
- `head -n fichero`
  presenta las `n` primeras lineas del fichero en pantalla
- `tail -n fichero`
  presenta las `n` últimas lineas del fichero en pantalla

Gestión de archivos

- `cp fichero1 fichero2`
  copia `fichero1` con el nombre `fichero2`.
- `mv fichero1 fichero2`
  mueve o renombra `fichero1` a `fichero2`.
- `rm fichero`
  borra el archivo.
- `mkdir directorio`
  crea un nuevo directorio.

Gestión de archivos

- `cp fichero1 fichero2`
  copia `fichero1` con el nombre `fichero2`.
- `mv fichero1 fichero2`
  mueve o renombra `fichero1` a `fichero2`.
- `rm fichero`
  borra el archivo.
- `mkdir directorio`
  crea un nuevo directorio.

Gestión de archivos

- `cp fichero1 fichero2`
  copia `fichero1` con el nombre `fichero2`.
- `mv fichero1 fichero2`
  mueve o renombra `fichero1` a `fichero2`.
- `rm fichero`
  borra el archivo.
- `mkdir directorio`
  crea un nuevo directorio.

Gestión de archivos

- `cp fichero1 fichero2`
  copia `fichero1` con el nombre `fichero2`.
- `mv fichero1 fichero2`
  mueve o renombra `fichero1` a `fichero2`.
- `rm fichero`
  borra el archivo.
- `mkdir directorio`
  crea un nuevo directorio.

Partición de archivos

- `split`
  Suponiendo que `fichero1` tiene 4500 lineas, la orden:

  ```
  split -l 1000 fichero1 fich
  ```

  generará 5 ficheros:

  `fichaa`, `fichab`, `fichac`,`fichad`, `fichae`.

  Los cuatro primeros contienen 1000 lineas, el último 500.

Partición de archivos

- `cut` selecciona determinadas columnas de un fichero.

```
1 alpha 2 azul
2 alpha 3 rojo
3 beta 3 rojo
4 gamma 2 rojo
```

```
cut -f 2,4 -d " " fic1
```

```
alpha azul
alpha rojo
beta rojo
gamma rojo
```

Concatenación de archivos por lineas

- `cat` permite también concatenar archivos.

  `cat fichero1>>fichero2`

  copia el contenido de fichero1 al final de fichero2

Concatenación de archivos por columnas

`join` concatena archivos por columnas. A partir de 2 ficheros `fic1` y `fic2` genera una serie de lineas uniendo las que tienen el mismo valor en un determinado campo (columna).

- `-1 n` donde `n` es la columna que actuará como campo clave en el primer fichero (por defecto es la primera).
- `-2 n` donde `n` es la columna que actuará como campo clave en el segundo fichero (por defecto es la primera).
- `-t c` donde `c` es el carácter que se utiliza como separador de campos (por defecto es el espacio).

Concatenación de archivos por columnas

`join` concatena archivos por columnas. A partir de 2 ficheros `fic1` y `fic2` genera una serie de lineas uniendo las que tienen el mismo valor en un determinado campo (columna).

Las opciones básicas para trabajar con `join` son:

- `-1 n` donde `n` es la columna que actuará como campo clave en el primer fichero (por defecto es la primera).
- `-2 n` donde `n` es la columna que actuará como campo clave en el segundo fichero (por defecto es la primera).
- `-t c` donde `c` es el carácter que se utiliza como separador de campos (por defecto es el espacio).

Concatenación de archivos por columnas

`join` concatena archivos por columnas. A partir de 2 ficheros `fic1` y `fic2` genera una serie de lineas uniendo las que tienen el mismo valor en un determinado campo (columna).

Las opciones básicas para trabajar con `join` son:

- `-1 n` donde `n` es la columna que actuará como campo clave en el primer fichero (por defecto es la primera).
- `-2 n` donde `n` es la columna que actuará como campo clave en el segundo fichero (por defecto es la primera).
- `-t c` donde `c` es el carácter que se utiliza como separador de campos (por defecto es el espacio).

Concatenación de archivos por columnas

`join` concatena archivos por columnas. A partir de 2 ficheros `fic1` y `fic2` genera una serie de lineas uniendo las que tienen el mismo valor en un determinado campo (columna).

Las opciones básicas para trabajar con `join` son:

- `-1 n` donde `n` es la columna que actuará como campo clave en el primer fichero (por defecto es la primera).
- `-2 n` donde `n` es la columna que actuará como campo clave en el segundo fichero (por defecto es la primera).
- `-t c` donde `c` es el carácter que se utiliza como separador de campos (por defecto es el espacio).

Concatenación de archivos por columnas

`join` concatena archivos por columnas. A partir de 2 ficheros `fic1` y `fic2` genera una serie de lineas uniendo las que tienen el mismo valor en un determinado campo (columna).

Las opciones básicas para trabajar con `join` son:

- `-1 n` donde `n` es la columna que actuará como campo clave en el primer fichero (por defecto es la primera).
- `-2 n` donde `n` es la columna que actuará como campo clave en el segundo fichero (por defecto es la primera).
- `-t c` donde `c` es el carácter que se utiliza como separador de campos (por defecto es el espacio).

Concatenación de archivos por columnas

`fic1`

1 Albacete 22
2 Alicante 32
3 Almería 33
4 Murcia 2

`fic2`

1 24
2 23
3 12
4 18

Concatenación de archivos por columnas

fic1

1 Albacete 22
2 Alicante 32
3 Almería 33
4 Murcia 2

fic2

1 24
2 23
3 12
4 18

```
join -1 1 -2 1 fic1 fic2
```

Concatenación de archivos por columnas

`fic1`

1 Albacete 22
2 Alicante 32
3 Almería 33
4 Murcia 2

`fic2`

1 24
2 23
3 12
4 18

```
join -1 1 -2 1 fic1 fic2
```

1 Albacete 22 24
2 Alicante 32 23
3 Almería 33 12
4 Murcia 2 18

Concatenación de archivos por columnas

`fic1`

1 Albacete 22
2 Alicante 32
3 Almería 33
4 Murcia 2

`fic2`

1 24
2 23
3 12
4 18

```
join -1 1 -2 1 fic1 fic2
```

1 Albacete 22 24
2 Alicante 32 23
3 Almería 33 12
4 Murcia 2 18

Las columnas que actúan como campo clave deben estar ordenadas

sed y grep

`grep patrón archivos`
donde el patrón a buscar es una expresión regular.
Devolverá todas las lineas de los archivos en los que aparece el `patrón`.
Este puede ser una cadena de texto o una expresión regular.

sort

```
4 Murcia 2
2 Alicante 32
1 Albacete 22
3 Almería 33
```

sort

```
4 Murcia 2
2 Alicante 32
1 Albacete 22
3 Almería 33
```

```
sort arch1
```

```
1 Albacete 22
2 Alicante 32
3 Almería 33
4 Murcia 2
```

sort

```
4 Murcia 2
2 Alicante 32
1 Albacete 22
3 Almería 33
```

```
sort arch1
```

```
1 Albacete 22
2 Alicante 32
3 Almería 33
4 Murcia 2
```

`sort -n arch2.num` ordena numéricamente.

Búsqueda de ayuda

BASH dispone de un programa para generar ayuda relativa a cualquier programa o comando del sistema (`man`).

```
man sort
```

mostrará en pantalla la ayuda de dicho programa.

Variables

Permiten dar generalidad a una simple lista de tareas:

Script:

```
x=10
echo $x
```

Resultado: `10`

Aritmética de variables

Script:

```
a=3
let b=$a+3
c=$(($a*3))
echo $a $b $c
```

Resultado: `3 6 9`

Aritmética con números reales con AWK

Script:

```
a=4;b=7
c=$(echo $a $b|awk 'print sqrt($1*$1+$2*$2)')
echo $a $b $c
```

Resultado: `7.416198`

Asignar a una variable el resultado de un proceso

Script:

```
x=$(seq 1 10)
echo $x
```

Resultado: `1 2 3 4 5 6 7 8 9 10`

Trabajando con textos

Script:

```
echo “Hola Mundo”
printf "Hola %s\n" mundo
```

Resultado:

```
Hola mundo
Hola mundo
```

Script:

```
interlocutor=mundo
echo Hola $interlocutor
```

Resultado: `Hola mundo`

Códigos de printf

%d	Número entero
%*n*d	Número entero formateado a *n* caracteres
%f	Número real
%*m*.*n*f	Número real con *n* decimales formateado a *m* caracteres
%s	Cadena de carácteres

Expresiones lógicas:Operadores numéricos

- Igual `-eq`
- No igual `-ne`
- Menor que `-lt`
- Menor o igual que `-le`
- Mayor que `-gt`
- Mayor o igual que `-ge`

Expresiones lógicas:Operadores de texto

- Igual =
- No igual !=
- Menor que <
- Mayor que >

Para utilizar el comando `test` con textos es necesario entrecomillar las variables:

Expresiones lógicas:Operadores de texto

Script:

```
a=Elefante;b=Cocodrilo
test "$a" = "$b"
echo $?
test "$a" != "$b"
echo $?
```

Resultado:

```
1
0
```

Expresiones lógicas: Encadenando condiciones

Podemos encadenar condiciones con los operadores `Y` lógico (`&&`), `O` lógico (`||`) y `NO` (`!`).

Script:

```
test "$a" != "$b" && test 2 -eq 2
echo $?
```

Resultado:

```
1
```

Expresiones lógicas: Encadenando condiciones

`||` tiene la misma precedencia que la suma y `&&` la misma que el producto, cuando sea necesario habrá que poner paréntesis:

Script:

```
a=Elefante;b=Cocodrilo
test "$a" == "$b" && test 2 eq 3 || test 2 eq 2
echo $?
test "$a" == "$b" && (test 2 -eq 3 || test 2 -eq 2)
echo $?
```

Resultado:

```
1
0
```

Operadores lógicos con ficheros

Existen diversos operadores para consultar características sobre los ficheros presentes en el sistema. Por ejemplo:

```
test -e mifichero.txt
echo $?
```

devolverá 0 si el fichero existe.

Arrays

Script:

```
declare -a identificador
identificador=(1 22 33 40 51)
echo ${identificador[3]}
```

Resultado:

```
40
```

Hay que tener en cuenta que:

- Son necesarias las llaves
- El primer elemento del array es el 0
- Si se sustituye el índice entre corchetes por un asterisco, devuelve todos los valores

Arrays

Script:

```
declare -a identificador
identificador=(1 22 33 40 51)
echo ${identificador[3]}
identificador[3]=50
echo $identificador[*]
```

Resultado:

```
40
1 22 33 50 51
```

como ves podemos modificar directamente los elementos de un array.

Concatenación de variables

Para concatenar dos variables de texto en BASH basta con escribirlas juntas tal como se puede ver en los siguientes ejemplos:

```
extension=txt;fichero=datos
echo $fichero.$extension
```

```
extension=txt;fichero=datos
fichero=${fichero}001.$extension
```

Si no resulta evidente donde termina el nombre de la variable es necesario delimitarlo explicitamente con llaves tal como se ve en el segundo ejemplo.

Concatenación de variables

El entrecomillado simple convertiría toda la concatenación en un literal:

Script:

```
extension=txt;fichero=datos
fichero='${fichero}001.$extension'
echo $fichero
```

Resultado: `${fichero}001.$extension`

Hacer ejecutable un script

En la primera linea del script:

```
#!/bin/bash
```

Hacer ejecutable el script:

```
chmod 755 miscript
```

Ejecutar el script:

```
./miscript
```

Parámetros que se pasan al programa

Suponiendo que el script `parametros` contiene:

```
#!/bin/sh
echo $3 $2 $1
```

La siguiente llamada:

```
$ parametros uno dos tres
```

producirá la siguiente salida:

```
tres dos uno
```

Pidiendo información al usuario

```
read algo
echo $algo
```

Puede utilizarse de forma más sofisticada añadiendo un prompt para que el usuario sepa que hacer:

```
read -p "Dime algo:  " -a algo
echo Has dicho $algo
```

Pidiendo información al usuario

Más interesante puede ser utilizar arrays en combinación con el comando `select` para generar menús para el usuario:

Pidiendo información al usuario

Más interesante puede ser utilizar arrays en combinación con el comando `select` para generar menús para el usuario:

script:
```
declare a acciones
acciones=(copiar renombrar borrar)
select p "Escoge:" accion in ${acciones[*]}
    do echo Has elegido $accion
    break
done
```

Pidiendo información al usuario

Más interesante puede ser utilizar arrays en combinación con el comando `select` para generar menús para el usuario:

script:
```
declare a acciones
acciones=(copiar renombrar borrar)
select p "Escoge:" accion in ${acciones[*]}
    do echo Has elegido $accion
    break
done
```

resultado:
```
1) copiar
        2) renombrar
        3) borrar
        #?  2
        Has elegido renombrar
```

Tuberias y redirecciones

```
ls -l>listado.txt
ls -l>>listado.txt
```

Tuberias y redirecciones

```
ls -l>listado.txt
ls -l>>listado.txt

cat archivo1 > archivo2
cat archivo1 >> archivo2
```

Tuberias y redirecciones

```
ls -l>listado.txt
ls -l>>listado.txt

cat archivo1 > archivo2
cat archivo1 >> archivo2

ls -l|more
ls|awk 'print $8'|grep 4
```

IF

```
if [ "$x" = "$k" ]; then
     echo Son iguales
else
     echo No son iguales
fi
```

El resultado variará dependiendo de si las variables son iguales o no.

La identación de lineas no es obligatoria pero ayuda a leer el programa, en los ejemplos que siguen aparece a menudo.

IF

Un ejemplo con variables numéricas

```
if [ $edad -le 18 ]; then
      echo Joven
else
      echo Mayor
fi
```

CASE

La herramienta `case` es más adecuada cuando son varias las opciones que se presentan al usuario:

```
case $edad in
     8|9|10|11|12|13) echo niño ;;
     14|15|16|17|18) echo joven ;;
     *) echo mayor;;
esac
```

Bucle FOR

Script:

```
for v in $(seq 1 10);do
      let v2=$v*2;
      printf "%d*%d=%d\n" $v 2 $v2
done
```

Resultado:

1*2=2
2*2=4
.....
9*2=18
10*2=20

Bucle WHILE

Script:

```
a=1
while test $a -le 10;do
      echo $a
      let a=$a+1
done
```

Resultado:

1
2
..
9
10

Bucle WHILE

Script:

```
for v in $(seq 1 10);do
      for v2 in $(seq 1 10); do
           let v3=$v*$v2;
           printf "%d*%d=%d\t" $v $v2 $v3
      done
      printf "\n"
done
```

Resultado:

Las tablas de multiplicar

Bucle UNTIL

Script:

```
usuario=pepe
until who|grep $usuario>/dev/null;do
      sleep 30
done
echo ...  y ahora lanzo el proceso
```

Resultado:

Hasta que no se conecte el usuario pepe no termina el bucle.

Break y Continue

Script:

```
for i in $(seq 1 5);do
      if test $i -eq 3;then
           break
      fi
      echo $i
done
```

Resultado:

```
1
2
3
```

Break y Continue

Script:

```
for i in $(seq 1 5);do
      if test $i -eq 3;then
           continue
      fi
      echo $i
done
```

Resultado:

```
1
2
4
5
```

Funciones y alias

```
listado () {ls -la;}
listado
```

Una función sin parámetros no resulta muy útil. En realidad para estos casos es más habitual utilizar el comando `alias`:

```
alias listado="ls -la"
listado
```

Funciones

```
function factorial(){
      f=1
      for i in $(seq 2 $1);do
           f=$(($f*$i))
      done
      echo $f
}

factorial 12
```

Funciones

El siguiente ejemplo introduce una función para esperar a la conexión de un usuario, en este caso el nombre del usuario se pasa como parámetro a la función:

```
esperar_a(){
      usuario=$1
      until who|grep $usuario>/dev/null;do
           sleep 5
      done
}
```