

---

# SMTA

Herramienta para el tratamiento de archivos.

25 de febrero 2018

Felix Aguilar Ferrer

Andreu Llabres Bañuls

---

## Índice

<b>Enunciado proyecto:</b>	<b>4</b>
<b>Base teórica:</b>	<b>4</b>
<b>Diseño solución:</b>	<b>6</b>
Diseño de la estructura de directorios:	6
Librerías:	7
Información almacenada:	10
Estructura de menús:	10
<b>Implementación de la solución:</b>	<b>14</b>
menu.sh (Felix Aguilar Ferrer):	14
control.sh (Felix Aguilar Ferrer):	17
order.sh (Andreu Llabres Bañuls):	23
smta.sh (Felix Aguilar Ferrer):	28
config.sh (Felix Aguilar Ferrer):	29
mor.sh (Andreu Llabres Bañuls):	31
sort.sh (Felix Aguilar Ferrer):	32
check.sh (Felix Aguilar Ferrer):	33
change.sh (Felix Aguilar Ferrer):	37
trash.sh (Felix Aguilar Ferrer):	38
installer.sh (Felix Aguilar Ferrer):	43
lib.sh (Felix Aguilar Ferrer):	45
install.sh (Felix Aguilar Ferrer):	50
uninstall.sh (Felix Aguilar Ferrer):	55
<b>Pruebas:</b>	<b>57</b>
Agrupación de archivos	57
Papelera de reciclaje	59
Configuración:	61
Sistema de lenguaje:	62
Instalador:	63
Desinstalador:	65

---

<b>Organización del proyecto:</b>	<b>66</b>
<b>Puntos de discusión:</b>	<b>66</b>
Elección del proyecto y división:	66
Abandono de miembros del grupo:	66
Propuestas del profesor:	67
<b>Problemas encontrados:</b>	<b>67</b>
Abandono de miembros del grupo:	67
Borrado masivo del sistema:	67
<b>Bibliografía:</b>	<b>68</b>

---

## Enunciado proyecto:

Es una herramienta para tratar archivos, estos se introducen por el directorio especificado como entrada y se ordenan en el directorio destinado como salida. Por cada archivo tratado se escribe una línea en el fichero que contiene todos los archivos tratados, para saber su ubicación en todo momento. En cualquier momento se ha de poder cambiar la configuración de la herramienta. Además ha de contener un sistema de papelera que permita enviar archivos a un directorio especial y poder ser recuperados en cualquier momento por el usuario.

Pasos a seguir:

1. Instalador y desinstalador de este.
2. Interfaz multi idioma (Castellano e inglés).
3. Sistemas de menús de tratamiento de los archivos.
  - a. Permitiendo la ordenación de archivo por:
    1. Primera letra.
    2. Última letra.
    3. Extensión.
  - b. Borrado de archivos y recuperación de estos.
4. Base de datos de los movimientos y de las opciones relacionadas con archivos.

## Base teórica:

Para realizar el trabajo se han usado algunas ordenes que no se han comentado en clase:

- rev: Permite invertir la salida estandard invirtiendo el orden de los caracteres del valor introducido. Ejemplo:

```
echo $i | rev
```

- uniq: Se encarga de quitar valores ya repetidos dentro de un archivo y para funcionar tiene que estar acompañado con un sort anteriormente. Ejemplo:

```
cat pass.txt | sort | uniq > pass-listos.txt
```

- sed: Es un editor de flujo para el filtrado y transformación de texto. Un editor de flujo es aquel que permite realizar operaciones básicas de transformación de texto en una entrada de flujo. Ejemplo:

```
sed -i "s/viejo string/nuevo string/g" archivo
```

Por lo tanto viendo la ejecución anterior, permite modificar un fichero mediante la patrones entre otras cosas.

- 
- awk: Es un intérprete para el lenguaje de programación AWK. Este es útil para la modificación de información almacenada en ficheros, recuperación y procesamiento de texto, y para la creación de prototipos y experimentación con algoritmos. Un programa AWK es una secuencia de patrones (acciones) pares y definiciones de funciones. Normalmente se usan para programas cortos en la consola de comandos encapsulados por " para evitar la interpretación por parte del shell.

```
awk -v req=patrón '$0 !~ req' archivo > temp && mv temp archivo
```

Por ejemplo en la ejecución anterior buscaría el patrón en el archivo y eliminaría la línea entera que contenga este patrón. para obtener un valor dentro de awk hay que usar la opción -v de la siguiente forma.

```
-v req=patrón
```

- arrays: Son variables que contienen más de un dato, estas se obtienen mediante un readarray:

```
readarray files < <(cat archivo)
```

Esto permitirá tener todas las líneas del fichero como un campo del array. Para obtener la longitud del array se utiliza:

```
int=${#files[@]}
```

Para obtener un valor en concreto del array el usuario ha de utilizar el siguiente código:

```
${files[indice]}
```

## Diseño solución:

### Diseño de la estructura de directorios:

La estructura de directorios una vez instalados que utiliza la herramienta es la siguiente:

▼ Includes	Hay dos partes, el instalador, este es el directorio SMTA y la herramienta SMTA la cual contiene los directorios Includes, Languages y Scripts.
▼ Data	
config.txt	
files.txt	
trash.txt	
▼ Lib	Herramienta SMTA:
control.sh	
menu.sh	
order.sh	
▼ Languages	Includes contiene los archivos de información ubicados en el directorio Data los cuales son config.txt, files.txt y trash.txt explicados más adelante. Además de todas las librerías utilizadas por la herramienta, también explicadas a más adelante.
eng.txt	
esp.txt	
▼ Scripts	Languages contiene todas las bases de información para los idiomas posibles en la herramienta.
change.sh	
check.sh	
config.sh	
mor.sh	
smta.sh	Finalmente Scripts contiene todos los archivos utilizados para los menús y acciones de la herramienta, incluyendo el archivo smta.sh, el cual es el archivo a ejecutar para utilizar la herramienta.
sort.sh	
trash.sh	
▼ SMTA	Instalador herramienta SMTA:
▼ Includes	
Archive.zip	
eng.txt	
esp.txt	
install.sh	
lib.sh	
uninstall.sh	
installer.sh	Este contiene la carpeta Includes que contiene todo lo necesario y los métodos de instalación y desinstalación de la herramienta, como archive.zip es el contenedor comprimido de la herramienta para facilitar su instalación. Además de los dos

archivos de idiomas y la librería, lib.sh, donde estan las funciones necesarias para el funcionamiento correcto del instalador, installer.sh, el cual es el primer fichero a ejecutar para la instalación de la herramienta.

---

## Librerías:

**menu.sh:** Aquí se contiene todas las funciones relacionadas con la creación de menús en la herramienta, aquí también se obtiene el lenguaje que se utiliza y el path al archivo de idiomas correcto. Esta librería contiene las siguientes funciones:

- divider: Permite escribir una línea divisoria, esta acepta parámetros pero no son necesarios:
  - \$1 = carácter para dividir.
  - \$2 = número de caracteres a escribir.
- title: Permite crear un apartado para el título con un divisor especial, el parámetro necesario es:
  - \$1 = valor identificativo del string.
- options: Permite escribir por pantalla una serie de valores con un número antes del texto o valor, los parámetros necesarios son:
  - \$1, \$2, ..., \$n = valor identificativo del string o bien una variable.
- input: Permite escribir por pantalla una línea de texto donde el usuario deberá de introducir una respuesta, el parámetro necesario es:
  - \$1 = valor identificativo del string.
- correct: Se utiliza para escribir por pantalla una línea subrayada de color verde, Los valores de entrada que permite son los siguientes:
  - \$1 = valor identificativo del string o bien una variable.
  - \$2 = indicador de si es una variable o bien un identificador de string.
- error: Se utiliza para escribir por pantalla una línea subrayada de color rojo, Los valores de entrada que permite son los siguientes:
  - \$1 = valor identificativo del string o bien una variable.
  - \$2 = indicador de si es una variable o bien un identificador de string.
- text: Se utiliza para escribir por pantalla una línea de texto. Los valores de entrada que permite son los siguientes:
  - \$1 = valor identificativo del string o bien una variable.
- output: Se utiliza para escribir por pantalla una línea subrayada de color azul con una variable, Los valores de entrada que permite son los siguientes:
  - \$1 = variable.
  - \$2 = valor identificativo del string, no es necesario.

---

**control.sh:** Esta librería es usada para modificar, acceder y eliminar el contenido de los archivos utilizados para almacenar información.

- **configs:** Esta función es utilizada para obtener la información del archivo de configuraciones el valor obtenido se guarda en una variable.
  - \$1 = nombre del campo a obtener.
  - \$2 = variable donde se guardará el valor obtenido.
- **strings:** Esta función es la encargada de obtener el valor correspondiente para los strings mostrados al usuario, este la devuelve por la variable \$str.
  - \$1 = identificador de la línea a devolver.
- **changedir:** Permite cambiar de directorio utilizado por la herramienta.
  - \$1 = Directorio nuevo.
  - \$2 = Directorio antiguo.
  - \$3 = Se mueven los archivos? necesita un 0 o un 1.
  - \$4 = Se mueven los anteriores path de los archivos dentro de la papelera? necesita un 0 o un 1.
- **addline:** Utilizado para crear una línea nueva en un fichero con divisor de campos ":".
  - \$1 = Ruta absoluta del fichero a modificar.
  - \$2, \$3, ..., \$n = campos a añadir.
- **delline:** Utilizado para eliminar una línea dentro de un archivo.
  - \$1 = patrón a buscar en el fichero.
  - \$2 = archivo donde buscar y eliminar la línea.
- **resetfiles:** Permite limpiar la base de datos de files.txt y mueve todos los ficheros a la carpeta de entrada, este no requiere ningún parámetro de entrada.
- **check:** Muestra por pantalla los datos que encaje con el patrón introducido en el fichero donde buscar.
  - \$1 = Fichero txt donde buscar el patrón.
  - \$2 = Patrón a buscar.
  - \$3 = Campo donde buscarlo.

**order.sh:** Esta librería permite la agrupación de los archivos por extensión, primer carácter y último carácter. Aparte de mover estos desde el directorio de entrada al de salida.

- **ordfiles:** Es la función principal para el tratado de los archivos.
  - \$1 = Método de ordenación que se quiere utilizar, sería, ext, fchar o bien lchar.



- 
- \$2 = La ruta que se utiliza para que el usuario introduzca los archivos a ser agrupados.
  - \$3 = La ruta donde se agruparan los archivos.
  - countp: Permite contar los puntos que tiene el nombre de un archivo con tado con la extensión.
    - \$1 = Hace referencia al tercer parámetro de entrada de la función principal.
  - get\_fc: Permite recoger el primer caracter de cada archivo.
    - \$1 = Hace referencia al tercer parámetro de entrada de la función principal.
  - get\_lc: Obtiene el último caracter de cada nombre de archivo.
    - \$1 = Hace referencia al tercer parámetro de entrada de la función principal.

**lib.sh:** Esta librería es usada en el instalador de la herramienta SMTA, contiene las funciones del menú y a parte de funciones de tratado de archivos.

- string: Esta función es la misma que la de la librería control.sh.
- divider: Esta función es la misma que la de la librería menu.sh.
- title: Esta función es la misma que la de la librería menu.sh.
- text: Esta función es la misma que la de la librería menu.sh.
- options: Esta función es la misma que la de la librería menu.sh.
- input: Esta función es la misma que la de la librería menu.sh.
- correct: Esta función es la misma que la de la librería menu.sh.
- error: Esta función es la misma que la de la librería menu.sh.
- output: Esta función es la misma que la de la librería menu.sh.
- addfile: Esta función permite añadir líneas de de base de datos a un fichero, sin embargo, si el fichero no existe se crea.
  - \$1 = Ruta absoluta del archivo.
  - \$2, \$3 ... \$n = No es necesaria para el funcionamiento de la función, cada una es un capo de la línea divididos por “:”.
- checkinput: Es un menú para confirmar que el valor introducido por el usuario es el que este quiere.
  - \$1 = Variable a verificar.
  - \$2 = Valor identificativo para el texto que acompaña la variable.
  - \$3 = valor identificativo para el texto del input.
  - \$4 = paso de la instalación actual.

### Información almacenada:

La información se guarda en archivos txt, los cuales se utilizan para almacenar información sobre los archivos tratados con la herramienta y la configuración de esta, estos son:

- config.txt: Este contiene la configuración de la herramienta, como el idioma, la ubicación de la instalación, el método de ordenación y los directorios de la papelera, input y output.
- files.txt: Contiene todos los archivos tratados por la herramienta de ordenación exceptuando los que se ubican en la papelera, cada línea representa un archivo siendo el primer campo el nombre y el segundo la ruta absoluta donde se encuentra el archivo.
- trash.txt: Este contiene los archivos ubicados en la papelera, como en el anterior cada línea es un archivo, siendo el primer campo el nombre y el segundo el directorio donde se encontraba antes de moverse a la papelera.

### Estructura de menús:

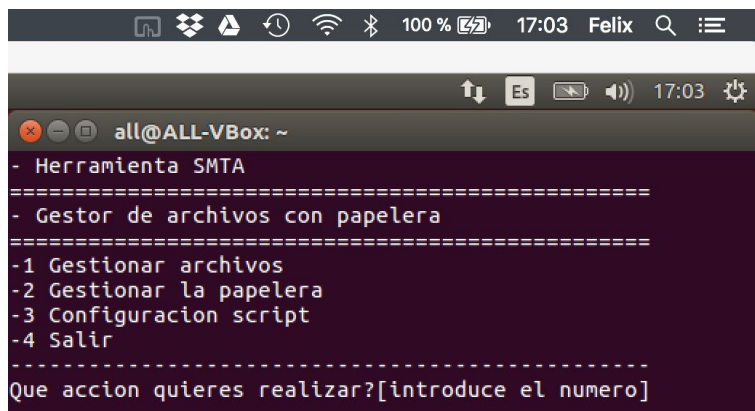
La herramienta tiene una estructura de menús para la interacción con el usuario la cual está pensada para facilitar la interacción con este.

Para poder ejecutar la herramienta se utiliza el archivo Scripts/smta.sh el cual se ubicará en el directorio de instalación impuesto por el usuario en el proceso de instalación explicado más adelante.

Al ejecutarlo aparecerá el siguiente menú obviamente dependiendo del idioma configurado este estará en inglés o en castellano.

Desde aquí se podrá acceder a los tres menús principales, gestionar archivos, gestionar la papelera y la configuración de la herramienta, para navegar por este menú se utiliza los números que

aparecen en el lateral de las opciones incluyendo el de salir de la herramienta.



```
all@ALL-VBox: ~  
- Herramienta SMTA  
=====
```

```
- Gestor de archivos con papelera  
=====
```

```
-1 Gestionar archivos  
-2 Gestionar la papelera  
-3 Configuracion script  
-4 Salir  
-----  
Que accion quieres realizar?[introduce el numero]
```

Ahora entraremos en el menú de gestión de archivos. En este se puede observar que aparece información sobre dónde y cómo va a trabajar la herramienta, incluyendo el directorio de entrada y el directorio de salida además del método de ordenación activo.

```
all@ALL-VBox: ~  
- Gestionar archivos  
=====
```

Directorio =	/media/sf_Ubuntu/output/
Directorio =	/media/sf_Ubuntu/input/
Método de ordenacion =	ext

```
-----  
-1 Ordena ahora  
-2 Revisar los archivos ordenados  
-3 Salir  
-----  
Que accion quieres realizar?[introduce el numero] █
```

Desde aquí se puede realizar una ordenación de los archivos introducidos en la carpeta de entrada utilizando el método por defecto, además de poder acceder a un menú el cual

muestra y revisa la base de datos de los archivos tratados.

Desde aquí se pueden realizar varias acciones para solventar problemas de la base de datos los cuales pueden afectar severamente la funcionalidad de la herramienta.

```
all@ALL-VBox: ~  
- Revisar los archivos ordenados  
=====
```

- Leyenda de los colores	
=====	
Archivos correctos	
Archivos que no han sido localizados	
-----	
/media/sf_Ubuntu/output/c/read.doc	
/media/sf_Ubuntu/output/g/ten.log	
/media/sf_Ubuntu/output/l/local.html	
/media/sf_Ubuntu/output/m/agua.com	
/media/sf_Ubuntu/output/t/hola.txt	
-----	
-1 Mirar por archivos extraviados en el directorio de entrada	
-2 Eliminar los archivos extraviados	
-3 Volver atras	
-----	
Que accion quieres realizar?[introduce el numero] █	

Por otro lado, está la gestión de la papelera, este menú permite eliminar un fichero o recuperarlo, dentro de estas dos acciones hay un sistema de menús simples para facilitar el trabajo al usuario explicados en las pruebas del script.

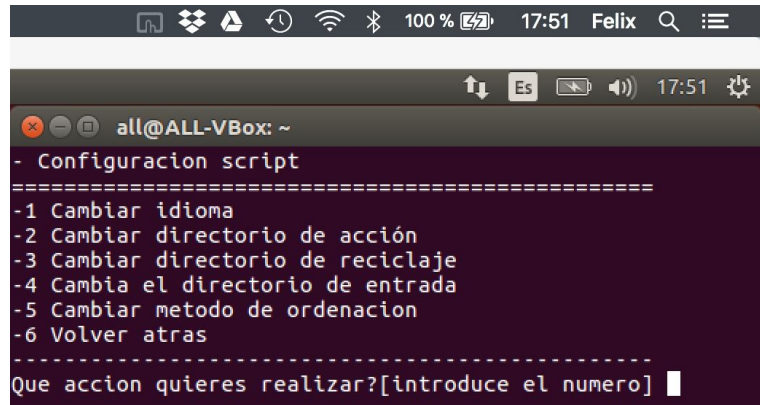
```
all@ALL-VBox: ~  
- Papelera  
=====
```

-1 Eliminar archivo
-2 Recuperar archivo
-3 Volver atras

```
-----  
Que accion quieres realizar?[introduce el numero] █
```

Finalmente está el menú de configuración en el cual se puede cambiar la configuración de la herramienta en cualquier momento.

Como se ve en la imagen, la configuración está dividida en 5 menús diferentes, cada uno con un submenú para realizar la configuración del parámetro al que hace referencia el menú, esto es así para evitar confusión por parte del usuario. Ahora se entrará en cada menú de la configuración.

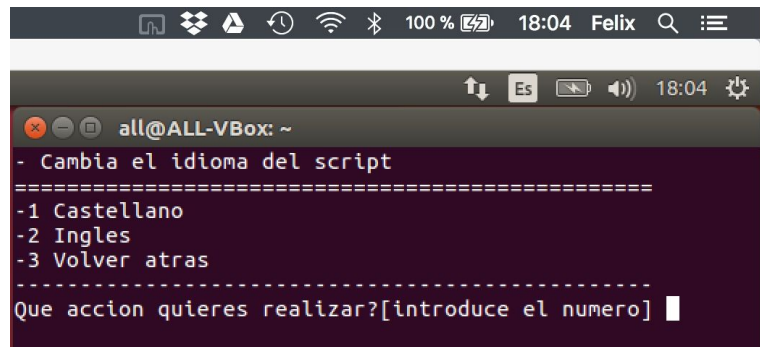


```
all@ALL-VBox: ~  
- Configuracion script  
=====
```

- 1 Cambiar idioma
- 2 Cambiar directorio de acción
- 3 Cambiar directorio de reciclaje
- 4 Cambia el directorio de entrada
- 5 Cambiar metodo de ordenacion
- 6 Volver atras

```
-----  
Que accion quieres realizar?[introduce el numero] █
```

Para cambiar el idioma se introduce el valor identificador de este y al momento el idioma será el que se haya introducido siempre y cuando no sea el mismo que estaba anteriormente.

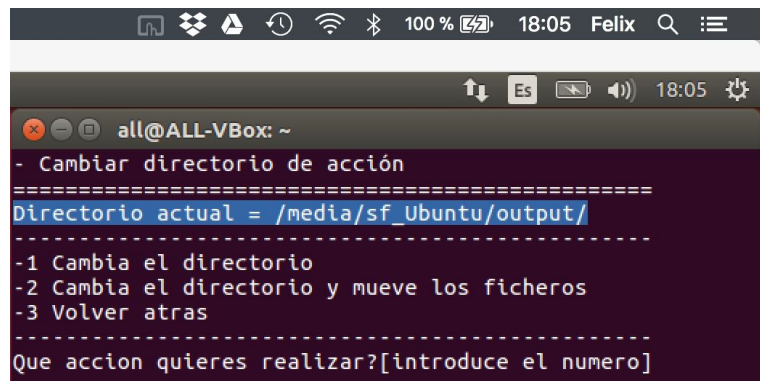


```
all@ALL-VBox: ~  
- Cambia el idioma del script  
=====
```

- 1 Castellano
- 2 Ingles
- 3 Volver atras

```
-----  
Que accion quieres realizar?[introduce el numero] █
```

Los menús para cambiar las ubicaciones de los directorios son los mismos en los tres directorios por lo que solamente se va explicar en uno de ellos. En el menú se puede observar cual es el directorio actual que se utiliza, y en las opciones para ejecutar permite mover solamente el directorio sin tocar los archivos o bien mover todo el conjunto a otro directorio.



```
all@ALL-VBox: ~  
- Cambiar directorio de acción  
=====
```

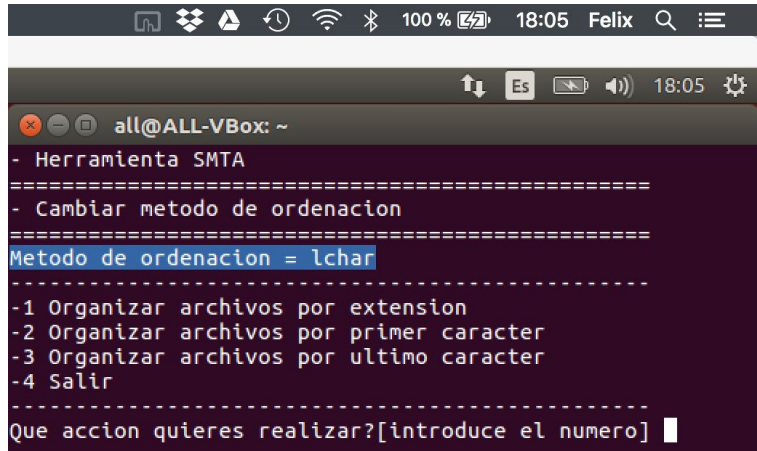
Directorio actual = /media/sf\_Ubuntu/output/

```
-----
```

- 1 Cambia el directorio
- 2 Cambia el directorio y mueve los ficheros
- 3 Volver atras

```
-----  
Que accion quieres realizar?[introduce el numero]
```

Finalmente está el menú para cambiar la ordenación por defecto del sistema, además al cambiar se reordena todos los archivos. Desde aquí se puede elegir entre los tres métodos para ordenar los archivos siendo por extensión, por primer carácter o por último carácter.



```
all@ALL-VBox: ~  
- Herramienta SMTA  
=====
```

```
- Cambiar metodo de ordenacion  
=====
```

```
Metodo de ordenacion = lchar  
-----
```

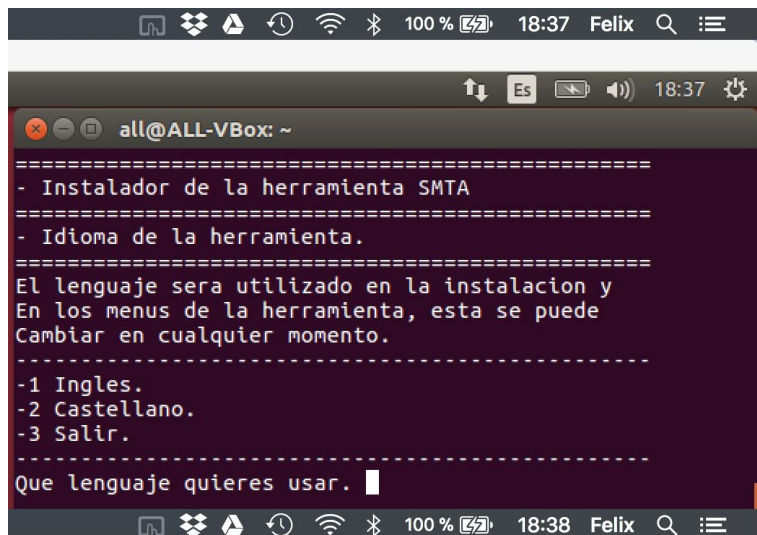
```
-1 Organizar archivos por extension  
-2 Organizar archivos por primer caracter  
-3 Organizar archivos por ultimo caracter  
-4 Salir  
-----
```

```
Que accion quieres realizar?[introduce el numero] █
```

La herramienta de instalación, también tiene una estructura de menús parecida a la de la herramienta SMTA:

Para ejecutar la herramienta de instalación, se ejecuta el archivo installer.sh el cual es el principal para la instalación.

Al ejecutarlo aparecerá el menú de la imagen en el cual se elige el idioma utilizado por el instalador.



```
all@ALL-VBox: ~  
=====
```

```
- Instalador de la herramienta SMTA  
=====
```

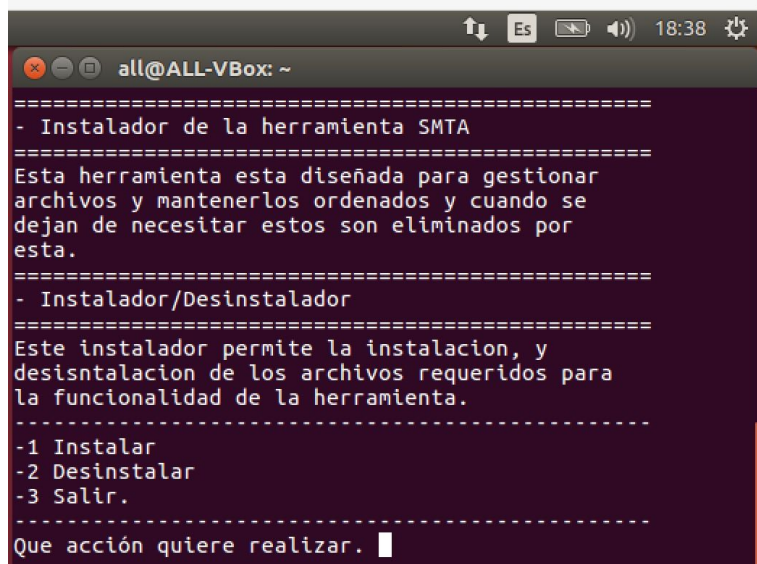
```
- Idioma de la herramienta.  
=====
```

```
El lenguaje sera utilizado en la instalacion y  
En los menus de la herramienta, esta se puede  
Cambiar en cualquier momento.  
-----
```

```
-1 Ingles.  
-2 Castellano.  
-3 Salir.  
-----
```

```
Que lenguaje quieres usar. █
```

Entonces aparecerá el menú principal para la instalación en el cual se puede elegir que acción a realizar, si es una instalación o bien una desinstalación. En cualquiera de las dos



```
all@ALL-VBox: ~  
=====
```

```
- Instalador de la herramienta SMTA  
=====
```

```
Esta herramienta esta diseñada para gestionar  
archivos y mantenerlos ordenados y cuando se  
dejan de necesitar estos son eliminados por  
esta.  
=====
```

```
- Instalador/Desinstalador  
=====
```

```
Este instalador permite la instalacion, y  
desinstalacion de los archivos requeridos para  
la funcionalidad de la herramienta.  
-----
```

```
-1 Instalar  
-2 Desinstalar  
-3 Salir.  
-----
```

```
Que acción quiere realizar. █
```



---

se proceden con una serie de pasos para completar la acción deseada por el usuario.

## Implementación de la solución:

### menu.sh (Felix Aguilar Ferrer):

```
#Felix Aguilar Ferrer.
#Libreria para la creación de menús.

#Llamamos a la librería de control y buscamos que language actual se
esta utilizando
#y creamos una variable para acceder a los ficheros de strings del
idioma adecuado.
. ${pathsmta}/Includes/Lib/control.sh
configs language leng
leng=${pathsmta}/Languages/$leng.txt

function divider(){
    #Felix Aguilar Ferrer.
    #Crea una división en el menú.
    #Inputs $1 = carácter para dividir.
    #      $2 = número de caracteres para la linea.

    if [ -z $1 ]
    then
        char="-"
    else
        char=$1
    fi

    if [ -z $2 ]
    then
        length=49
    else
        length=$2
    fi

    txt=""
    while [ $length -gt 0 ]
    do
        txt=$txt$char
        let length=length-1
    done
    echo $txt

    #Se eliminan las variables usadas.
    unset txt
    unset length
    unset char
}

function title (){
    #Felix Aguilar Ferrer.
    #Crea el titulo del menu, este incorpora un divider.
```

---

```

#Inputs $1 = valor identificativo del string, ej: s30.

#La función string obtiene el valor del fichero de strings adecuado.
string $1
echo - $str

#Se eliminan las variables que ya no se necesitan.
unset str

#Se crea una división.
divider "="
}

function options (){
    #Felix Aguilar Ferrer.
    #Crea el abanico de opciones, este incorpora un divider.
    #Inputs $1 $2 $3 ... = valor identificativo del string, ej: s30 o
    bien una variable.

    #Se crea un índice y se obtiene la cantidad de inputs de la orden.
    n=1
    i=$#

    #Mientras n sea menor que i, se irá imprimiendo las opciones.
    while [ $n -le $i ]
    do

        #La función string obtiene el valor del fichero de strings
        adecuado.
        string $1

        #Si no se ha encontrado ningun string posible con el valor
        introducido,
        #Se escribirá el valor de la variable introducida.
        if [ $? = '0' ]
        then
            echo -$n $str
        else
            echo -$n $1
        fi
        shift

        #Se le añade 1 a la variable n
        let n=n+1
    done

    #Se eliminan las variables que ya no se necesitan.
    unset n
    unset str
    unset i

    #Se crea una división.
    divider
}

function input(){

```

---

```

#Felix Aguilar Ferrer.
#Crea una pregunta para que el usuario introduzca un valor
#Inputs $1 = valor identificativo del string, ej: s30.
#Output $ans = valor introducido por el usuario.

#La función string obtiene el valor del fichero de strings adecuado.
string $1
echo -n "$str "

#Se eliminan las variables que ya no se necesitan.
unset str

#Se incorpora el valor introducido en la variable.
read ans
}

function correct(){
#Felix Aguilar Ferrer.
#Crea una línea de texto de color verde.
#Inputs $1 = valor a imprimir o string.
#      $2 = para saber si es un string o un valor ej: 0 o 1.

#Se compara el valor del input $2 para saber que se ha introducido
en el $1.
if [ $2 -eq 1 ]
then
    str=$1
else

    #La función string obtiene el valor del fichero de strings
adecuado.
    string $1
fi
echo -e "\e[42m$str\e[49m"

#Se eliminan las variables que ya no se necesitan.
unset str
}

function error(){
#Felix Aguilar Ferrer.
#Crea una línea de texto de color rojo.
#Inputs $1 = valor a imprimir o string.
#      $2 = para saber si es un string o un valor ej: 0 o 1.

#Se compara el valor del input $2 para saber que se ha introducido
en el $1.
if [ $2 -eq 1 ]
then
    str=$1
else

    #La función string obtiene el valor del fichero de strings
adecuado.
    string $1
fi

```



---

```

        echo -e "\e[41m$str\e[49m"

        #Se eliminan las variables que ya no se necesitan.
        unset str
    }

function text(){
    #Felix Aguilar Ferrer.
    #Crea una línea de texto.
    #Inputs $1 = string.

    #La función string obtiene el valor del fichero de strings adecuado.
    string $1
    echo $str

    #Se eliminan las variables que ya no se necesitan.
    unset str
}

function output(){
    #Felix Aguilar Ferrer.
    #Crea una línea de texto de color azul.
    #Inputs $1 = variable a imprimir por pantalla.
    #          $2 = valor identificativo del string, ej: s30.

    #Se observa si se ha introducido el valor de string.
    if [ ! -z $2 ]
    then

        #La función string obtiene el valor del fichero de strings
        adecuado.
        string $2
        echo -e "\e[44m$str = $1\e[49m"
        else

        #Si no se ha introducido, se escribe por pantalla la variable
        sola.
        echo -e "\e[44m$1\e[49m"
        fi

        #Se eliminan las variables que ya no se necesitan.
        unset str
    }
}

```

#### control.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer.
#Libreria para la obtención de parámetros en ficheros.

function configs(){
    #Felix Aguilar Ferrer.
    #Recoge un campo del archivo config.txt y lo guarda en la variable
    deseada.
    #Inputs $1 = Campo del archivo.
    #          $2 = Variable donde se guarda.
}

```

---

```

        read $2 < <(grep $1 ${pathsmta}/Includes/Data/config.txt | cut -d
        ':' -f 2)
    }

function string(){
    #Felix Aguilar Ferrer.
    #Recoge el string del archivo de lenguaje indicado y lo guarda en la
    variable str.
    #Inputs $1 = String a buscar.

    read str < <(grep $1 $leng | cut -d ':' -f 2)
}

function changedir(){
    #Felix Aguilar Ferrer.
    #Cambia el directorio especificado que este dentro de config.txt.
    #Inputs $1 = Nuevo directorio.
    #      $2 = Directorio viejo.
    #      $3 = Se mueven los archivos? 0 o 1.
    #      $4 = Se mueve papeleria? 0 o 1.

    #Se realizan comprobaciones para evitar errores por parte del
    usuario.
    if [ ! "$1" = "" ]
    then
        if [ ! $1 = $2 ]
        then
            if [ ! -d $1 ]
            then

                #Si se puede crear el directorio, entonces se cambia en
                el archivo config.txt.
                mkdir $1
                if [ $? = 0 ]
                then
                    sed -i "s|:$2:|:$1:g"
                    ${pathsmta}/Includes/Data/config.txt
                    correct s37 0

                #Si se han de mover los archivos, se realiza un cp
                de todos los archivos y
                #directorios que contiene el directorio anterior.
                if [ $3 -eq 1 ]
                then
                    cp -r $2* $1

                #Si ha sido un éxito, se pasa a procesar la base
                de datos de archivos.
                if [ $? = 0 ]
                then

                    #Se elimina la carpeta anterior.
                    rm -r $2

                    #Se genera un array con todos los valores de
                    files.txt.

```

---

```

                                readarray a <
${pathsmta}/Includes/Data/files.txt

                                #Se crea un index y la longitud de los
arrays.
                                i=0
                                int=${#a[@]}

                                #Se van a procesar todas las entradas del
array.
                                while [ $i -lt $int ]
                                do
                                    #Se lee el nombre del archivo y se busca
por el sistema sabiendo aproximadamente
                                    #la ruta donde se encuentra ($1) y
recortando el nombre.
                                    read name < <(echo ${a[$i]} | cut -d':'
-f 1)
                                    read where < <(find $1 -name "$name" 2>
/dev/null | rev | cut -d '/' -f2- | rev)

                                    #Si ha sido exitosa la búsqueda, se
realiza lo siguiente.
                                    if [ $? = 0 ]
                                    then

                                        #Se recupera su antiguo path.
                                        read old < <(echo ${a[$i]} | cut
-d':' -f 2)

                                        #Se sustituye el path antiguo con el
nuevo.
                                        sed -i "s|:$old:|:$where/|g"
${pathsmta}/Includes/Data/files.txt
                                    fi

                                    #Se suma uno al índice y se eliminan las
variables.
                                    let i=$i+1
                                    unset where
                                    unset old
                                    unset name
                                done

                                #Se eliminan las variables que no se
utilizan.
                                unset a
                                unset int
                                unset i

                                #Tratamiento de la papelera.
                                if [ $4 = 1 ]
                                then

                                    #Se genera un array con todos los
valores de files.txt.

```

```

readarray a <
${pathsmta}/Includes/Data/trash.txt

#Se crea un index y la longitud de los
arrays.
i=0
int=${#a[@]}

while [ $i -lt $int ]
do
    #Se van a procesar todas las
    entradas del array.
    read old < <(echo $a[$i] | cut
    -d':' -f 2 )
    read dir < <(echo $a[$i] | cut
    -d':' -f 2 | cut -d '/' -f5- )

    #Se sustituye el path antiguo con el
    nuevo.
    sed -i "s|:$old:|:$1$dir:|g"
    ${pathsmta}/Includes/Data/trash.txt

    #Se suma uno al índice y se eliminan
    las variables.
    let i=$i+1
    unset dir
    unset old
done

#Se eliminan las variables que no se
utilizan.
unset a
unset int
unset i
fi

correct s36 0
else
    error s35 0
fi

fi
else
    error s34 0
fi

else
    error s33 0
fi

else
    error s32 0
fi

else
    error s31 0
fi
}

function addline(){

```

---

```

#Felix Aguilar Ferrer.
#Permite la inserción de líneas con el carácter divisor :.
#Inputs $1 = Fichero destino.
#      $2 $3 $4 ... = Campos a insertar.

#Obtenemos el path y movemos el puntero.
pathfile=$1
shift

#Se crea el índice y el número de parámetros introducidos se obtiene
#además de crear el inicio del string.
i=$#
n=1
txt=""

#El bucle para añadir campos.
while [ $n -le $i ]
do
txt="$txt$1:"
shift
let n=$n+1
done

#Se introduce la línea al archivo.
echo -e $txt 1>> $pathfile
}

function delline(){
#Felix Aguilar Ferrer.
#Permite eliminar una línea con un parámetro.
#Inputs $1 = patrón a buscar.
#      $2 = fichero donde buscar y eliminar.

#Se elimina la línea donde ha encontrado el patrón.
awk -v req=$1 '$0 !~ req' $2 > temp && mv temp $2
}

function resetfiles(){
#Felix Aguilar Ferrer
#Mueve todos los ficheros al directorio input y
#elimina la entradas de estos en el fichero files.txt.

#Se obtiene los path del directorio de entrada y salida.
. ${pathsmta}/Includes/Lib/control.sh
configs new input
configs directory out

#Se recogen todas las entradas del fichero files.txt
readarray files < <(cat ${pathsmta}/Includes/Data/files.txt)

#Se recoge la longitud del array además de crear un índice para
procesarlos.
int=${#files[@]}
i=0

#Si no esta vacío, longitud = 0, no se realizará nada más.

```

---

```

    if [ ! $int -eq 0 ]
    then

        #Se procesa cada entrada en el fichero.
        while [ $i -lt $int ]
        do

            #Se obtienen el nombre y la ruta del archivo.
            read name < <(echo ${files[$i]} | cut -d ':' -f 1)
            read path < <(echo ${files[$i]} | cut -d ':' -f 2)

            #Se mueve el archivo al fichero de entrada y se elimina la
entrada del fichero files.txt
            mv $path$name $input
            delline $name ${pathsmta}/Includes/Data/files.txt

            #Se aumenta el indice en 1 y se eliminan las variables.
            let i=i+1
            unset name
            unset path
        done

        #Finalmente se eliminan los directorios restantes en el
directorio de salida.
        rm -r $out/*

        #Se eliminan las variables utilizadas.
        unset out
        unset files
        unset input
    fi
}

function check(){
    #Felix Aguilar Ferrer
    #Para buscar y mostrar los archivos por pantalla.
    #Inputs $1 = Fichero txt donde busca los nombres.
    #        $2 = Valor a buscar dentro de este.
    #        $3 = Campo donde busca el nombre.

    #Se guardan en un array todos los valores que coincidan con el dato
introducido.
    readarray files < <(cut -d ':' -f $3 $1 | grep "$2")

    #Comprobamos que el array no está vacío.
    if [ ! -z $files ]
    then
        #Si no lo esta, se obtiene la longitud y se genera un índice.
        int=${#files[@]}
        i=0

        #Entramos en un bucle el cual guardará los datos en forma de
lista en una variable.
        while [ $i -lt $int ]
        do
            inside="$inside ${files[$i]}"
        done
    fi
}

```

```

        let i=$i+1
        done

        #Se generará una lista de los datos impuestos.
        options $inside

        #Se elimina la variable inside.
        unset inside
    else

        #Se genera este error cuando no se encuentran coincidencias.
        error s53 0
        divider
    fi
}

```

#### order.sh (Andreu Llabres Bañuls):

```

# Creado por Andreu Llabres Bañuls
. ${pathsmta}/Includes/Lib/control.sh
. ${pathsmta}/Includes/Lib/menu.sh
# $1 Método de ordenación ej: ext , fchar, lchar o datf
# $2 Directorio de entrada enter por defecto
# $3 Directorio de salida Test por defecto

# Function para contar los puntos.
function countp(){
    # Recoge los archivos que contengan punto.
    ls -p ${1}*,* | rev | cut -d / -f 1 | rev | sort > ${1}sk/br.txt
    countt=$(cat ${1}sk/br.txt | wc -l) #Contar las líneas del archivo.
    a=1
    # $x vale cada línea del archivo.
    for x in `cat ${1}sk/br.txt`
    do
        i=1
        num=0
        var=$(expr length $x) # Cuenta los caracteres de $x.
        while [ $i -le $var ]; do
            # Mientras puedes comprobar que . existe
            if [ `expr substr $x $i 1` = "." ]; then
                # Sumale .
                let num=num+1
            fi
            # Posición.
            i=`expr $i + 1`
        done
        if [ $a -le $countt ]; then # Se ejecuta mientras $a no llegue
al total de líneas.
            let num=num+1
            # Lee línea por línea a partir del último punto.
            cat ${1}sk/br.txt | head -$a | tail -1 | cut -d . -f $num
>> $1sk/ext0.txt
            let a=a+1
        fi
    done
    # Ordena y quita las extensiones repetidas.
}

```

```

cat ${1}sk/ext0.txt | sort > ${1}sk/ext.txt
# Borrar variables usadas.
unset i
unset a
unset num
}
# Funcion para sacar el primer caracter de cada archivo.
function get_fc(){
    # Recoge el nombre de los archivos.
    ls -p ${1} | grep -v / | sort > ${1}sk/br.txt
    for x in `cat ${1}sk/br.txt`
    do
        # Recoge el primer carácter.
        var=$(expr substr $x 1 1)
        #Lo que vale $var lo mete en un archivo
        echo $var >> ${1}sk/fc1.txt
    done
    # Ordena los caracteres repetido
    cat ${1}sk/fc1.txt | sort >> ${1}sk/fc.txt
}
# Funcion para sacar el último caracter de cada archivo.
function get_lc(){
    # Recoge el nombre de los archivos invertidos
    ls -p ${1} | grep -v / | rev > ${1}sk/br.txt
    for a in `cat ${1}sk/br.txt`
    do
        # Recoge el primer carácter.
        var=$(expr substr $a 1 1)
        #Lo que vale $var lo mete en un archivo
        echo $var >> ${1}sk/lc1.txt
    done
    # Ordena los caracteres repetidos
    cat ${1}sk/lc1.txt | sort >> ${1}sk/lc.txt
}
# Funcion Principal
function ordfiles(){
    pathb=$3
    if [[ ! -d ${3}/sk/ ]]; then
        # Comprueba si existen archivos en el directorio de entrada.
        read lecint < <(ls -p ${2} | grep -v / | sort)
        # Si el archivo arcp.txt no está vacío.
        if [[ ! "$lecint" = '' ]]; then
            # Directorio sk es donde se almacena toda las bases de datos que
            se maneja en la búsqueda de archivos.
            mkdir ${3}sk
            # mueve el Contenido de directorio de entrada a el de salida
            mv ${2}* ${3}
            # Comprueba si existen archivos.
            ls -p ${3} | grep -v / | sort > ${3}sk/arcp.txt
            # Si el archivo arcp.txt no está vacío.
            if [ -s ${3}sk/arcp.txt ]; then
                if [[ ${1} = ext ]]; then
                    countp $pathb
                    # Recoge los nombres de los archivos con su extensión.
                    ls -p ${3}*.* | rev | cut -d / -f 1 | rev | sort >
                    ${3}sk/arc.txt

```



```

# For para mostrar las extensiones de los archivos.
for i in `cat ${3}sk/ext.txt`
do
    count=0
    #Genera los directorios dependiendo de la extensión.
    if [[ ! -d ${3}$i/ ]]; then
        mkdir ${3}$i
    fi
    #For para mostrar el nombre del archivo con su
extensión
    for x in `cat ${3}sk/arc.txt`
    do
        #Si el archivo acaba en .extensión
        if [[ "$x" = *.$i ]]; then
            # Cambiar el nombre si es igual que el del
directorio en el que se mueve.
            if [[ "$x" = $i.* ]]; then
                # Mueve el archivo al directorio creado
para el.

                mv ${3}$x ${3}$i/$count-xarchivex.$i
                # Añade una línea nueva al archivo
files.txt

                addline
${pathsmta}/Includes/Data/files.txt $count-xarchivex.$i ${3}$i/
                # Borra la línea que es usada para el
for, ya que sino hacemos esto siempre valdrá lo mismo.
                delline $x ${3}sk/arc.txt
                let count=count+1
            else
                # Mueve el archivo al directorio creado
para el.

                mv ${3}$x ${3}$i/$x
                # Añade una línea nueva al archivo
files.txt

                addline
${pathsmta}/Includes/Data/files.txt $x ${3}$i/
                # Borra la línea que es usada para el
for, ya que sino hacemos esto siempre faldrá lo mismo.
                delline $x ${3}sk/arc.txt
                let count=count-1
            fi
        fi
    done
done
#Ver si quedan archivos existentes que no tengan
extensión.

read lec < <(ls -p ${3} | grep -v / | sort)
#Si no está vacío.
if [ ! "$lec" = '' ]; then
    ls -p ${3} | grep -v / | sort > ${3}sk/arcsext.txt
    if [[ -d ${3}Unknown/ ]]; then
        pathu=Unknown/
        for a in `cat ${3}sk/arcsext.txt`
        do
            # Mueve el archivo al directorio creado para
el.

```

```

mv ${3}$z ${3}$pathu$a
# Añade una línea nueva al archivo files.txt
addline ${pathsmta}/Includes/Data/files.txt

$a ${3}$pathu/
# Borra la línea que es usada para el for,
ya que sino hacemos esto siempre valdrá lo mismo.
delline $x ${3}sk/arcsext.txt
done
else
pathu=Unknown/
mkdir ${3}$pathu
for z in `cat ${3}sk/arcsext.txt`
do
# Mueve el archivo al directorio creado para
el.
mv ${3}$z ${3}$pathu$z
# Añade una línea nueva al archivo files.txt
addline ${pathsmta}/Includes/Data/files.txt

$z ${3}$pathu/
# Borra la línea que es usada para el for,
ya que sino hacemos esto siempre valdrá lo mismo.
delline $x ${3}sk/arcsext.txt
done
fi
fi
# Borrar variables
unset count
unset lec
unset pathu
correct s47 0
elif [[ ${1} = fchar ]]; then
get_fc $pathb
# Recoge los nombres de los archivos con o sin
extensión.
ls -p ${3} | rev | cut -d / -f 1 | rev | sort >
${3}sk/arc.txt
# For para mostrar las extensiones de los archivos.
for i in `cat ${3}sk/fc.txt`
do
#Genera los directorios dependiendo de la extensión.
if [[ ! -d ${3}$i/ ]]; then
mkdir ${3}$i
fi
#For para mostrar el nombre del archivo con su
extensión
for x in `cat ${3}sk/arc.txt`
do
#Si el archivo acaba en .extensión
if [[ "$x" = $i* ]]; then
# Mueve el archivo al directorio creado para
el y si da un error no lo muestra.
mv ${3}$x ${3}$i/$x 2>/dev/null
# Añade una línea nueva al archivo files.txt
addline ${pathsmta}/Includes/Data/files.txt

$x ${3}$i/

```

```

                                # Borra la linea que es usada para el for,
ya que sino hacemos esto siempre valdrá lo mismo.
                                delline $x ${3}sk/arc.txt
                                fi
                                done
                                done
                                correct s63 0
                                elif [[ ${1} = lchar ]]; then
                                get_lc $pathb
                                # Recoge los nombres de los archivos con o sin
extensión.
                                ls -p ${3} | rev | cut -d / -f 1 | sort > ${3}sk/arc.txt
                                # For para mostrar las extensiones de los archivos.
                                for i in `cat ${3}sk/lc.txt`
                                do
                                #Genera los directorios.
                                if [[ ! -d ${3}${i}/ ]]; then
                                mkdir ${3}${i}
                                fi
                                #For para mostrar el nombre del archivo.
                                for x in `cat ${3}sk/arc.txt`
                                do
                                #
                                read iol < <(echo $x)
                                #Si el archivo comienza por
                                if [[ "$iol" = $i* ]]; then
                                # Giras el nombre del archivo
                                read ool < <(echo ${iol} | rev)
                                # Cambiar el nombre si es igual que el y si
da un error no lo muestra.
                                mv ${3}${ool} ${3}${i}/${ool} 2>/dev/null
                                # Añade una línea nueva al archivo files.txt
                                addline ${pathsmta}/Includes/Data/files.txt
                                ${ool} ${3}${i}/
                                # Borra la linea que es usada para el for,
ya que sino hacemos esto siempre valdrá lo mismo.
                                delline $x ${3}sk/arc.txt
                                fi
                                done
                                done
                                correct s64 0
                                fi
                                # Borrar variables
                                unset lecint
                                unset pathb
                                unset var
                                unset ool
                                unset iol
                                fi
                                # Eliminar base de datos
                                rm -r ${3}sk
                                else
                                #Error en no has introducido nada en el input.
                                error s14 0
                                fi
                                fi

```

---

```
}
```

### smta.sh (Felix Aguilar Ferrer):

```
#Felix Aguilar Ferrer.
#Menu de principal de la herramienta, primera hoja de ejecución.

#Al ser una herramienta no queremos que aparezca el prompt.
clear

#Obtenemos la ruta de este archivo y recortamos el nombre y el primer
directorio "Scripts" y guardamos la ruta en la variable pathsmta.
read pathsmta < <( echo $(dirname $(readlink -f $0)) | rev | cut -d'/'
-f2- | rev)

#Obtenemos la librería del menú.
. ${pathsmta}/Includes/Lib/menu.sh

#El bucle se monta para estar en el menú hasta que se decida salir.
menu=0
while [ $menu -eq 0 ]
do

    #Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
    title s3
    title s4
    options s12 s13 s2 s15
    input s16
    clear

    #Se trata la entrada del usuario.
    case $ans in
        1)

            #Se envía al usuario al menú de ordenación.
            . ${pathsmta}/Scripts/sort.sh
            ;;

        2)
            #Working...
            . ${pathsmta}/Scripts/Trash.sh
            ;;

        3)
            #Se envía al usuario al menú de configuración.
            . ${pathsmta}/Scripts/config.sh
            ;;

        4)
            #Para salir del script.
            menu=1
            ;;

        *)
            error s19 0
            ;;
    esac
done
unset menu
```

---

### config.sh (Felix Aguilar Ferrer):

```
#Felix Aguilar Ferrer.
#Menu para la configuración del script.

#Obtenemos la librería del menú.
. ${pathsmta}/Includes/Lib/menu.sh

function inputch(){
    #Felix Aguilar Ferrer.
    #Menú para cambiar el directorio donde se ordenan los archivos.
    #Inputs $1 = Para realizar el mover todos los archivos al nuevo
    directorio. [0 / 1]
    #          $2 = Cambiar path de los archivos dentro de la papelera? [1
    / 0]

    #Se pide el nuevo path del directorio y nombre.
    divider
    unset ans
    input s39

    #Se ejecuta la función para cambiar directorio dependiendo de la
    entrada.
    changedir $ans $olddir $1 $2
}

function chdir(){
    #Felix Aguilar Ferrer.
    #Menú para cambiar el directorio donde se ordenan los archivos.
    #Inputs $1 = Que directorio se cambia de nombre, este tiene que
    estar en el archivo config.txt.
    #          $2 = Frase para el Título.
    #          $3 = Cambiar path de los archivos dentro de la papelera? 1 o
    0

    menu=0
    while [ $menu -eq 0 ]
    do

        #Obtenemos la librería del menú.
        . ${pathsmta}/Includes/Lib/menu.sh

        #Obtenemos la librería de control.
        . ${pathsmta}/Includes/Lib/control.sh

        #Se obtiene el valor del directorio a cambiar.
        configs $1 olddir

        #Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
        title $2
        output $olddir s28
        divider
        options s29 s30 s7
        input s16
        clear
```

---

```

        #Se trata la entrada del usuario.
        case $ans in
            1)
                inputch 0 0
                ;;
            2)
                inputch 1 $3
                ;;
            3)
                menu=1
                ;;
            *)
                error s19 0
                ;;
        esac

done
menu=0
}

#El bucle se monta para estar en el menú hasta que se decida salir.
menu=0
while [ $menu -eq 0 ]
do

    #Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
    title s2
    options s8 s9 s10 s38 s11 s7
    input s16
    clear

    #Se trata la entrada del usuario.
    case $ans in
        1)

            #Accedemos al menú de cambio de idioma.
            . ${pathsmta}/Scripts/change.sh
            ;;

        2)
            #Accedemos al menú de cambio de directorio.
            chdir directory s9 1
            ;;

        3)
            #Accedemos al menú de cambio de directorio.
            chdir trash s10 0
            ;;

        4)
            #Accedemos al menú de cambio de directorio.
            chdir new s38 0
            ;;

        5)
            #Accedemos al menú de ordenación.
            . ${pathsmta}/Scripts/mor.sh
            ;;

        6)

```

```

        #Para salir del menú, imponemos 1 a la variable menu.
        menu=1
        ;;
    *)
        error s19 0
        ;;
esac
done
menu=0

```

### mor.sh (Andreu Llabres Bañuls):

```

#Creado por Andreu Llabres Bañuls.
#Librerías a utilizar.
. ${pathsmta}/Includes/Lib/menu.sh
. ${pathsmta}/Includes/Lib/order.sh
. ${pathsmta}/Includes/Lib/control.sh
# funcion para obtener el directorio de entrada, de salida y el tipo
agrupación por defecto.
configs directory direc
configs new new
configs orderby orderby
#El bucle se monta para estar en el menú hasta que se decida salir.
menu=0

while [ $menu -eq 0 ]
do
    #Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
    title s3
    title s11
    configs orderby orderby
    output $orderby s22
    divider
    options s44 s45 s46 s15
    input s16
    clear

    #Es la entrada que el usuario utilizara.
    case $ans in
        1)
            # Ordenar por extensión.
            if [ $orderby != ext ]; then
                resetfiles
                sed -i "s/:$orderby:/:ext:/g"
                ${pathsmta}/Includes/Data/config.txt
                ordfiles ext $new $direc
            else
                # Error ya esta en uso este tipo de agrupación.
                error s65 0
            fi
            ;;
        2)
            #Ordenar por primer carácter.
            if [ $orderby != fchar ]; then

```

```

        resetfiles
        sed -i "s/:$orderby:/:fchar:/g"
    ${pathsmta}/Includes/Data/config.txt
        ordfiles fchar $new $direc
    else
        error s65 0
    fi
;;

```

3)

```

    #Ordenar por el último carácter.
    if [ $orderby != lchar ]; then
        resetfiles
        sed -i "s/:$orderby:/:lchar:/g"
    ${pathsmta}/Includes/Data/config.txt
        ordfiles lchar $new $direc
    else
        error s65 0
    fi
;;

```

4)

```

    #Para salir del script.
    menu=1
;;

```

\*)

```

    error s19 0
;;

```

```

esac
done
menu=0

```

### sort.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer.
#Menu para el gestor de archivos.

#Obtenemos la librería del menú.
. ${pathsmta}/Includes/Lib/menu.sh
#Obtenemos la librería del control.
. ${pathsmta}/Includes/Lib/control.sh
#Obtenemos la librería de la ordenación de archivos.
. ${pathsmta}/Includes/Lib/order.sh

#El bucle se monta para estar en el menú hasta que se decida salir.
menu=0
while [ $menu -eq 0 ]
do

    #Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
    title s12

    #Se obtiene el directorio donde se hace la ordenación.

```



---

```

    configs directory direc
    output $direc s21

    #Se obtiene el directorio donde se introducen los ficheros para la
ordenación.
    configs new input
    output $input s21

    #Se obtiene el método de la ordenación.
    configs orderby order
    output $order s22
    divider
    options s20 s23 s15
    input s16

    clear
    case $ans in
        1)
            #Ordenar archivos por la opción por defecto
            ordfiles $order $input $direc
            ;;
        2)
            #Se redirige al menu para revisar los archivos en el
sistema.
            . ${pathsmta}/Scripts/check.sh
            ;;
        3)
            #Para salir del menú, imponemos 1 a la variable menu.
            menu=1
            ;;
        *)
            error s19 0
            ;;
    esac

    #Se eliminan las variables que ya no se necesitan.
    unset direc
    unset input
    unset order

done
menu=0

```

#### check.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer.
#Menú para revisar los ficheros dentro del sistema (utiliza files.txt).

#Obtenemos la librería del menú.
. ${pathsmta}/Includes/Lib/menu.sh

#El bucle se monta para estar en el menú hasta que se decida salir.
menu=0
while [ $menu -eq 0 ]
do

```

---

```

#Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
title s23
title s40
correct s41 0
error s42 0
divider
#Ahora se buscará y mostrará los archivos en files.txt

    #Se genera un array con el contenido del archivo files.txt.
    readarray a < ${pathsmta}/Includes/Data/files.txt

    #Se obtiene el directorio de la papelera.
    configs trash bins

    #Se obtiene la longitud total del array y se crea un índice.
    int=${#a[@]}
    i=0

    #En el bucle se tratará cada archivo.
    while [ $i -lt $int ]
    do
        #Se obtiene el path y el nombre del archivo correspondiente.
        read paths < <(echo ${a[$i]} | cut -d':' -f 2)
        read names < <(echo ${a[$i]} | cut -d':' -f 1)

        #Si existe en la ubicación de files.txt se mostrará de color
verde, de
        #lo contrario se mostrará de color rojo.
        if [ -f $paths$names ]
        then
            correct $paths$names 1
        else
            error $paths$names 1
        fi

        #Se le añade 1 al índice.
        let i=i+1
    done

    #Se eliminan las variables que nos son utilizadas.
    unset paths
    unset names
    unset i
    unset a
    unset int
    unset bin

divider
options s26 s24 s7
input s16
clear

#Se trata la entrada del usuario.
case $ans in

```

---

```

1)
#Para buscar los archivos en el directorio de entrada.

#Se genera un array con el contenido del archivo files.txt.
readarray a < ${pathsmta}/Includes/Data/files.txt

#Se obtiene el valor de la carpeta por donde se introducen los
archivos al sistema.
configs new news

#Se obtiene la longitud total del array y se crea un índice.
int=${#a[@]}
i=0

#Se genera una variable para comprobar la función.
res=0

#En el bucle se tratará cada archivo.
while [ $i -lt $int ]
do

    #Se obtiene el path y el nombre del archivo correspondiente.
    read paths < <(echo ${a[$i]} | cut -d':' -f 2)
    read names < <(echo ${a[$i]} | cut -d':' -f 1)

    #Si existe el archivo en la carpeta de input, se moverá.
    if [ -f $news/$names ]
    then
        mv $news/$names $paths
        res=1
    fi

    #Se incrementa el índice.
    let i=$i+1
done

#Se muestra el mensaje dependiendo del resultado de la
operación.
if [ $res -eq 1 ]
then
    correct s25 0
else
    error s27 0
fi

#Se borran las variables que no se necesitan.
unset news
unset a
unset i
unset res
unset paths
unset names
unset int
;;
2)

```

---

```

        #Para eliminar las entradas del fichero files.txt que no se
        ubican en la ordenación.

        #Se genera un array con el contenido del archivo files.txt.
        readarray a < ${pathsmta}/Includes/Data/files.txt

        #Se obtiene la longitud total del array y se crea un índice.
        int=${#a[@]}
        i=0

        #Se genera una variable para comprobar la función.
        res=0

        #En el bucle se tratará cada archivo.
        while [ $i -lt $int ]
        do

            #Se obtiene el path y el nombre del archivo correspondiente.
            read paths < <(echo ${a[$i]} | cut -d':' -f 2)
            read names < <(echo ${a[$i]} | cut -d':' -f 1)

            #Si no existe, se borra la entrada del archivo files.txt.
            if [ ! -f $paths$names ]
            then
                delline $names ${pathsmta}/Includes/Data/files.txt
                res=1
            fi

            #Se incrementa el índice.
            let i=$i+1
        done

        #Se muestra el mensaje dependiendo del resultado de la
operación.
        if [ $res -eq 1 ]
        then
            correct s25 0
        else
            error s27 0
        fi

        #Se borran las variables que no se necesitan.
        unset a
        unset paths
        unset names
        unset res
        unset i
        unset int
        ;;
3)

        #Para salir del menú, imponemos 1 a la variable menu.
        menu=1
        ;;
*)
        error s19 0

```

---

```
;;
esac
done
menu=0
```

#### change.sh (Felix Aguilar Ferrer):

```
#Felix Aguilar Ferrer.
#Cambio de idioma dels script.

function chlanguage(){
    #Felix Aguilar Ferrer.
    #Para cambiar el idioma en la base de datos.
    #Inputs $1 = idioma nuevo.

    #Se obtiene el language actual.
    configs language chlen

    #Si es el mismo error, sino cambia el idioma de la base de datos
    config.txt.
    if [ "$chlen" = "$1" ]
    then
        error s18 0
    else
        sed -i "s/:$chlen:/:$1:/g" ${pathsmta}/Includes/Data/config.txt
        correct s17 0
    fi
    unset chlen
}

#El bucle se monta para estar en el menú hasta que se decida salir.
menu=0
while [ $menu -eq 0 ]
do
    #Obtenemos la librería del menú.
    . ${pathsmta}/Includes/Lib/menu.sh

    #Se monta el menu, para ver más accede a /Includes/Lib/menu.sh.
    title s1
    options s5 s6 s7
    input s16

    #Comprobamos que acción ha pedido el usuario.
    clear
    case $ans in
        1)
            chlanguage esp
            ;;
        2)
            chlanguage eng
            ;;
        3)
            #Para salir del menú, imponemos 1 a la variable menu.
            menu=1
            ;;
        *)
```

---

```

        error s19 0
        ;;
    esac
done
menu=0

```

#### trash.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer
#Script para eliminar archivos dentro de la herramienta.

#Obtenemos las librerías del menu y control.
. ${pathsmta}/Includes/Lib/menu.sh
. ${pathsmta}/Includes/Lib/control.sh

#Obtenemos el path de la papelera.
configs trash trash

function delete (){
    #Felix Aguilar Ferrer
    #Muestra el menu para mover los archivos a la papelera:

    #Se genera el menu para eliminar el archivo deseado.
    menu=0
    while [ $menu = 0 ]
    do

        #Primera entrada.
        title s49
        title s50
        options s7
        input s48

        clear

        case $ans in
            1)
                menu=1
                ;;
            *)

                #Si ans no está vacío, entramos a la ejecución.
                if [ ! $ans = '' ]
                then

                    #Segunda entrada.
                    title s49
                    title s50

                    #Funcion para obtener los ficheros que coincidan con
                    el patrón dado.

                    check ${pathsmta}/Includes/Data/files.txt $ans 1
                    text s54
                    divider
                    input s48
                    clear

```

---

```

        #Si el usuario introduce un identificador.
        if [ ! $ans = "" ]
        then

                #Se obtiene el valor del identificador del array
y se obtiene el valor al cual apunta.
                let ans=ans-1
                file=${files[$ans]}

                #Menu de confirmación del borrado.
                title s49
                title s50
                output $file
                divider
                input s55
                clear
                case $ans in
                y)

                        #Se obtiene la ruta del fichero a borrar.
                        read paths < <(grep $file
${pathsmta}/Includes/Data/files.txt | cut -d':' -f 2)

                        #Si existe el archivo se procede a moverse.
                        if [ -e $paths$file ]
                        then
                                #Se mueve el fichero al directorio
especificado.

                                mv $paths$file $trash

                                #Se funciona, se modificaran las bases
de datos.

                                if [ $? = 0 ]
                                then

                                        #Se elimina la entrada que pertoca.
                                        delline $file
${pathsmta}/Includes/Data/files.txt

                                        #Se genera la linea nueva.
                                        addline
${pathsmta}/Includes/Data/trash.txt $file $paths
                                        correct s57 0

                                        fi
                                else
                                        error s62 0
                                fi

                                #Se eliminan las variables utilizadas.
                                unset paths
                                unset file
                                unset files
                                ;;
                n)
                        error s56 0

```

---

```

        *)
        ;;
    esac
    fi
    unset files
fi
;;
esac
done
menu=0
}

function recover (){
    #Felix Aguilar Ferrer
    #Muestra el menú para recuperar los archivos de la papelera:

    #Se genera el menú para recuperar el archivo deseado.
    menu=0
    while [ $menu = 0 ]
    do

        #Primera interacción con el usuario.
        title s49
        title s51
        options s7
        input s52

        clear

        case $ans in
            1)
                menu=1
                ;;
            *)

                #Si el usuario introduce un patrón se buscará por este.
                if [ ! $ans = '' ]
                then

                    #Segundo menu de interaccion con el usuario.
                    title s49
                    title s51

                    #Funcion para obtener los ficheros que coincidan con
                    el patrón dado.

                    check ${pathsmta}/Includes/Data/trash.txt $ans 1
                    text s54
                    divider
                    input s61
                    clear

                    #Si el usuario introduce un identificador.
                    if [ ! $ans = "" ]
                    then

```



---

```

        #Se obtiene el valor del identificador del array
y se obtiene el valor al cual apunta.
        let ans=ans-1
        file=${files[$ans]}

        #Menú de confirmación del borrado.
        title s49
        title s51
        output $file
        divider
        input s58
        clear
        case $ans in
        y)

                #Se obtiene la ruta del fichero a borrar.
                read paths < <(grep $file
${pathsmta}/Includes/Data/trash.txt | cut -d':' -f 2)

                #Si existe el archivo se procede a moverse.
                if [ -e $trash$file ]
                then

                        #Se mueve el fichero al directorio
especificado.

                        mv $trash$file $paths 2> /dev/null

                        #Se funciona, se modifican las bases de
datos.

                        if [ $? = 0 ]
                        then

                                #Se elimina la entrada que pertoca.
                                delline $file
                                ${pathsmta}/Includes/Data/trash.txt

                                #Se genera la linea nueva.
                                addline
                                ${pathsmta}/Includes/Data/files.txt $file $paths
                                correct s60 0
                                else

                                        #Si no se puede mover el archivo al
directorio original, este puede ser movido a la entrada.
                                        menu=0
                                        while [ $menu = 0 ]
                                        do

                                                #Muestra el mensaje de error y
pide la confirmacion para moverlo

                                                #a la entrada si no es así no se
realiza ninguna acción.

                                                error s66 0
                                                input s67
                                                case $ans in
                                                y)

```

---

```

                                #Se obtiene la ruta a la
entrada.                                configs new input

                                #Se mueve el fichero al
directorio especificado.                mv $trash$file $input

                                #Se elimina la entrada
que pertoca.                            delline $file

                                ${pathsmta}/Includes/Data/trash.txt

                                correct s60 0
                                menu=1
                                ;;
                                n)
                                error s59 0
                                menu=1
                                ;;
                                *)
                                error s19 0
                                ;;
                                esac
                                done
                                menu=0
                                fi
                                else
                                error s62 0
                                fi

                                #Se eliminan las variables utilizadas.
                                unset paths
                                unset file
                                unset files
                                ;;
                                n)
                                error s59 0
                                ;;
                                *)
                                ;;
                                esac
                                fi
                                unset files
                                fi
                                ;;
                                esac
                                done
                                menu=0
                                menu=0
                                menu=0
                                }

                                menu=0
                                while [ $menu = 0 ]
                                do

```

---

```

#Menu para seleccionar la acción a realizar.
title s49
options s50 s51 s7
input s16

clear

case $ans in
    1)
        delete
        ;;
    2)
        recover
        ;;
    3)
        menu=1
        ;;
    *)
        error s19 0
        ;;
esac
done
menu=0

```

#### installer.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer.
#Instalador de la utilidad.

#Obtenemos la ruta de este archivo para ubicar todos los requeridos para
la instalación.
read pathinst < <( echo $(dirname $(readlink -f $0)) | rev | cut -d'/'
-f2- | rev)

#El idioma por defecto.
leng=esp

#Generamos el menu para las opciones se utiliza la librería
menuinstall.sh.
. ${pathinst}/SMTA/Includes/lib.sh

#Se elimina el prompt de pantalla.
clear

#Menu para la selección de lenguaje de la instalación
menu=0
while [ $menu -eq 0 ]
do
    #Menu para la selección del idioma para el instalador.
    divider =
    title s0
    title s2
    text s24
    text s25
    text s26
    divider

```

---

```

options s3 s4 s5
input s6

clear

#Según el dato introducido por el usuario se selecciona el lenguaje
del instalador.
case $ans in
    1)
        leng=eng
        menu=1
        ;;
    2)
        leng=esp
        menu=1
        ;;
    3)
        clear
        error s23 0
        exit
        ;;
    *)
        error s7 0
        ;;
esac
done

clear

#Menu para la selección de la acción a realizar desinstalar o bien
instalar.
menu=0
while [ $menu -eq 0 ]
do
    divider =
    title s0
    text s34
    text s35
    text s36
    text s37
    divider =
    title s38
    text s39
    text s40
    text s41
    divider
    options s44 s45 s5
    input s43

    clear

    case $ans in
        1)
            . ${pathinst}/SMTA/Includes/install.sh
            ;;
        2)

```

---

```

        . ${pathinst}/SMTA/Includes/uninstall.sh
        ;;
    3)
        menu=1
        ;;
    *)
        error s7 0
        ;;
esac
done

#Se eliminan las variables usadas.
unset menu
unset leng
unset language
unset pathinst
unset inspath
unset order
unset inpath
unset outpath
unset trashpath
unset step

```

#### lib.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer
#Libreria del instalador.

function string(){
    #Felix Aguilar Ferrer.
    #Recoge el string del archivo de lenguaje indicado y lo guarda en la
    variable str.
    #Inputs $1 = String a buscar.
    read str < <(grep $1 ${pathinst}/SMTA/Includes/$leng.txt | cut -d
    ':' -f 2)
}

function divider(){
    #Felix Aguilar Ferrer.
    #Crea una división en el menú.
    #Inputs $1 = carácter para dividir.
    #      $2 = número de caracteres para la linea.

    if [ -z $1 ]
    then
        char="-"
    else
        char=$1
    fi

    if [ -z $2 ]
    then
        length=49
    else
        length=$2
    fi
}

```

---

```

        txt=""
        while [ $length -gt 0 ]
        do
            txt=$txt$char
            let length=length-1
        done
        echo $txt

        #Se eliminan las variables usadas.
        unset txt
        unset length
        unset char
    }

function title (){
    #Felix Aguilar Ferrer.
    #Crea el titulo del menu, este incorpora un divider.
    #Inputs $1 = valor identificativo del string, ej: s30.

    #La función string obtiene el valor del fichero de strings adecuado.
    string $1
    echo - $str

    #Se eliminan las variables que ya no se necesitan.
    unset str

    #Se crea una división.
    divider "="
}

function text(){
    #Felix Aguilar Ferrer.
    #Crea una línea de texto.
    #Inputs $1 = string.

    #La función string obtiene el valor del fichero de strings adecuado.
    string $1
    echo $str

    #Se eliminan las variables que ya no se necesitan.
    unset str
}

function options (){
    #Felix Aguilar Ferrer.
    #Crea el abanico de opciones, este incorpora un divider.
    #Inputs $1 $2 $3 ... = valor identificativo del string, ej: s30.

    #Se crea un índice y se obtiene la cantidad de inputs de la orden.
    n=1
    i=$#

    #Mientras n sea menor que i, se irá imprimiendo las opciones.
    while [ $n -le $i ]
    do

```

---

```

        #La función string obtiene el valor del fichero de strings
adecuado.
        string $1
        echo -n $str
        shift

        #Se le añade 1 a la variable n
        let n=n+1
    done

    #Se eliminan las variables que ya no se necesitan.
    unset n
    unset str
    unset i

    #Se crea una división.
    divider
}

function input(){
    #Felix Aguilar Ferrer.
    #Crea una pregunta para que el usuario introduzca un valor
    #Inputs $1 = valor identificativo del string, ej: s30.
    #Output $ans = valor introducido por el usuario.

    #La función string obtiene el valor del fichero de strings adecuado.
    string $1
    echo -n "$str "

    #Se eliminan las variables que ya no se necesitan.
    unset str

    #Se incorpora el valor introducido en la variable.
    read ans
}

function correct(){
    #Felix Aguilar Ferrer.
    #Crea una línea de texto de color verde.
    #Inputs $1 = valor a imprimir o string.
    #      $2 = para saber si es un string o un valor ej: 0 o 1.

    #Se compara el valor del input $2 para saber que se ha introducido
en el $1.
    if [ $2 -eq 1 ]
    then
        str=$1
    else

        #La función string obtiene el valor del fichero de strings
adecuado.
        string $1
    fi
    echo -e "\e[42m$str\e[49m"
}

```

---

```

        #Se eliminan las variables que ya no se necesitan.
        unset str
    }

    function error(){
        #Felix Aguilar Ferrer.
        #Crea una línea de texto de color rojo.
        #Inputs $1 = valor a imprimir o string.
        #      $2 = para saber si es un string o un valor ej: 0 o 1.

        #Se compara el valor del input $2 para saber que se ha introducido
        en el $1.
        if [ $2 -eq 1 ]
        then
            str=$1
        else

            #La función string obtiene el valor del fichero de strings
            adecuado.
            string $1
            fi
            echo -e "\e[41m$str\e[49m"

            #Se eliminan las variables que ya no se necesitan.
            unset str
        }

        function output(){
            #Felix Aguilar Ferrer.
            #Crea una línea de texto de color azul.
            #Inputs $1 = valor identificativo del string, ej: s30.
            #      esta mal ! $2 = para saber si es un string o un valor ej: 0
            o 1.

            #Se compara el valor del input $2 para saber que se ha introducido
            en el $1.
            if [ ! -z $2 ]
            then

                #La función string obtiene el valor del fichero de strings
                adecuado.
                string $2
                echo -e "\e[44m$str = $1\e[49m"
                else
                echo -e "\e[44m$1\e[49m"
                fi

                #Se eliminan las variables que ya no se necesitan.
                unset str
            }

            function addfile(){
                #Felix Aguilar Ferrer
                #Permite crear un fichero de texto que se usara para guardar
                información.
                #Inputs $1 = Ruta absoluta del fichero.

```



---

```

#          $2 ... $n = Campos de texto por línea

#Obtenemos el path y movemos el puntero.
pathfile=$1
shift

#Si no existe se crea el fichero
if [ ! -e $pathfile ]
then
    touch $pathfile
fi

if [ ! -z $1 ]
then
    #Se crea el índice y el número de parámetros introducidos se
obtiene
    #además de crear el inicio del string.
    i=$((i+1))
    n=1
    txt=""

    #El bucle para añadir campos.
    while [ $n -le $i ]
    do
        txt="$txt$1:"
        shift
        let n=n+1
    done

    #Se introduce la línea al archivo.
    echo -e $txt 1>> $pathfile

    unset i
    unset n
    unset txt

fi

unset pathfile
}

function checkinput(){
    #Felix Aguilar Ferrer.
    #Permite poner una condicion para verificar la input del usuario.
    #Inputs $1 = variable a verificar.
    #          $2 = texto que acompaña la variable.
    #          $3 = Texto input.
    #          $4 = step actual.

    #Se suma 1 al step actual.
    let a=a+1

    #Si el step es igual al futuro step entonces.
    if [ $step -eq $a ]
    then
        output $1 $2
    fi
}

```

---

```

        input $3

        clear

        case $ans in
        y)
            correct s28 0
            ;;
        n)
            step=$4
            ;;
        *)
            error s7 0
            step=$4
            ;;
        esac

    fi
unset a
}

```

#### install.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer
#Son los menús y acciones para la instalación de la herramienta.

menu=0
step=1
while [ $menu -eq 0 ]
do
    #Selección del lenguaje de la herramienta.
    if [ $step -eq 1 ]
    then
        function incase(){
            step=2
            language=$1
            title s1
            title s2
            checkinput $language s2 s27 1
        }

        title s1
        title s2
        text s24
        text s25
        text s26
        divider
        options s3 s4 s5
        input s6

        clear

        case $ans in
        1)
            incase eng
            ;;
        2)

```

```

        incase esp
        ;;
    3)
        menu=1
        ;;
    *)
        error s7 0
        ;;
esac
fi

#Selección del directorio de la herramienta.
if [ $step -eq 2 ]
then
    title s1
    title s8
    text s29
    text s30
    text s31
    divider
    options s9
    input s10

    clear

    case $ans in
        1)
            step=1
            ;;
        *)
            if [ ! $ans = '' ]
            then
                if [ -d $ans ]
                then
                    inspath=$ans
                    step=3
                    title s1
                    title s8
                    checkinput $inspath s8 s27 2
                else
                    error s11 0
                fi
            else
                error s11 0
            fi
            ;;
    esac
fi

#Selección del metodo de ordenacion.
if [ $step -eq 3 ]
then
    function incase(){
        step=4
        order=$1
        title s1

```

---

```

title s12
checkinput $order s12 s27 3
}
title s1
title s12
text s32
text s33
divider
options s14 s15 s16 s9
input s13

clear

case $ans in
    1)
        incase "ext"
        ;;
    2)
        incase "fchar"
        ;;
    3)
        incase "lchar"
        ;;
    4)
        step=2
        ;;
    *)
        error s7 0
        ;;
esac
fi

#Selección del directorio de entrada.
if [ $step -eq 4 ]
then
    title s1
    title s18
    text s47
    text s48
    divider
    options s9
    input s10

    clear

    case $ans in
        1)
            step=3
            ;;
        *)
            if [ -d $ans ]
            then
                inpath=$ans
                step=5
                title s1
                title s18

```

---

```

                checkinput $inpath s18 s27 4
            else
                error s11 0
            fi
        ;;
    esac
fi

#Selección del directorio de ordenación.
if [ $step -eq 5 ]
then
    title s1
    title s19
    text s51
    text s52
    divider
    options s9
    input s10

    clear

    case $ans in
        1)
            step=4
            ;;
        *)
            if [ -d $ans ]
            then
                outpath=$ans
                step=6
                title s1
                title s19
                checkinput $outpath s19 s27 5
            else
                error s11 0
            fi
        ;;
    esac
fi

#Selección del directorio de la basura.
if [ $step -eq 6 ]
then
    title s1
    title s20
    text s49
    text s50
    divider
    options s9
    input s10

    clear

    case $ans in
        1)
            step=5

```

---

```

        ;;
    *)
        if [ -d $ans ]
        then
            trashpath=$ans
            step=7
            title s1
            title s20
            checkinput $trashpath s20 s27 6
        else
            error s11 0
        fi
        ;;
    esac
fi

#Confirmación de los parámetros introducidos.
if [ $step -eq 7 ]
then
    title s1
    title s21
    output $language s2
    output $inspath s8
    output $order s12
    output $inpath s18
    output $outpath s19

    output $trashpath s20
    divider
    input s22

    clear

    case $ans in
        y)
            step=8
            ;;
        n)
            menu=1
            error s23 0
            ;;
        *)
            error s7 0
            ;;
    esac
fi

#Se realiza la instalación.
if [ $step -eq 8 ]
then
    text s53
    unzip ${pathinst}/SMTA/Includes/Archive.zip -d $inspath

    #Se monta el archivo de configuración y se crea la base de
    datos de archivos.
    addfile ${inspath}/Includes/Data/config.txt installdir $inspath

```

---

```

        addfile ${inspath}/Includes/Data/config.txt language $language
        addfile ${inspath}/Includes/Data/config.txt directory $outpath
        addfile ${inspath}/Includes/Data/config.txt new $inpath
        addfile ${inspath}/Includes/Data/config.txt trash $trashpath
        addfile ${inspath}/Includes/Data/config.txt orderby $order
        addfile ${inspath}/Includes/Data/files.txt
        addfile ${inspath}/Includes/Data/trash.txt

        menu=1
        correct s53 0
    fi
done
menu=0

```

#### unistall.sh (Felix Aguilar Ferrer):

```

#Felix Aguilar Ferrer
#Desinstalador de la herramienta SMTA.

menu=0
step=1
while [ $menu -eq 0 ]
do

    #Menu para la introducción del path de instalación.
    title s1
    title s56
    options s9
    input s57

    clear

    case $ans in
        1)
            menu=1
            ;;
        *)

            #Si no esta en blanco se utiliza como path.
            if [ ! $ans = '' ]
            then

                #Se comprueba que exista el directorio.
                if [ -d $ans ]
                then

                    #Se comprueba que exista los fichero principal de la
herramienta.
                    if [ -e ${ans}/Scripts/smta.sh ]
                    then

                        #Si existe entonces se eliminan los ficheros.
                        rm -r ${ans}/*

                        #Se comprueba que se ha podido realizar.
                        if [ $? = 0 ]

```

---

```
        then
            correct s58 0
        else
            error s59 0
        fi
    else
        error s60 0
    fi
else
    error s11 0
fi
;;
esac
done
menu=0
```

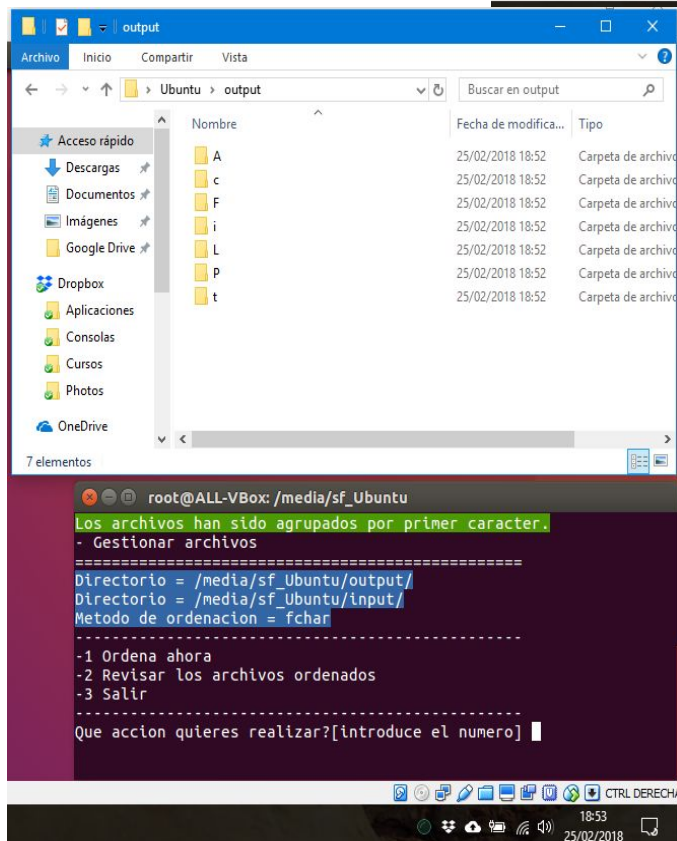
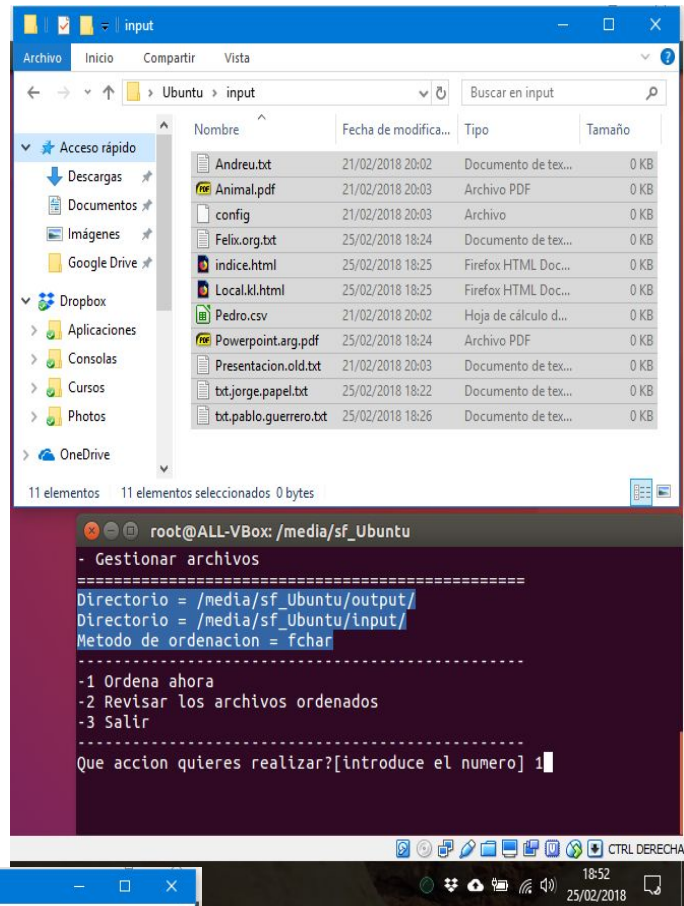


## Pruebas:

### - Agrupación de archivos

Sí se accede desde gestionar archivos a Ubuntu y desde windows como máquina externa para ver los cambios.

Ahora habría que desde el software pulsar 1 para ordenar por el método que este marcado, por defecto el método es elegido por el usuario.



Se puede ver que ha movido todo los archivos y los a agrupado dependiendo de su primer carácter y devuelve un mensaje de que si ha ido bien todo es de color verde.

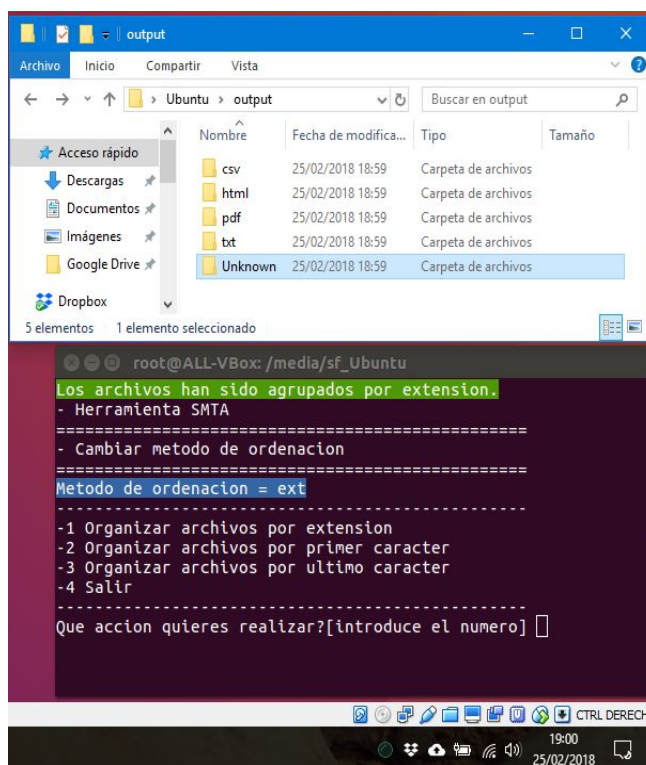
Para comprobar el estado de estos archivos si pulsas 2 en el menú en cual se estaba antes, se puede ver el estado actual de cada uno de los archivos. En el caso de que uno se extravié o el usuario decida sacarlo saldrá marcado como rojo ese archivo y el directorio en el que permanecía.

También se puede borrar estos archivos extraviados pulsado 2 en este menú y ya no aparecería en la base de archivos ordenados.

Si se pulsa en el método de ordenación que ya existe, se te advierte con un mensaje en rojo indicando que ya están agrupados por este tipo de ordenación.

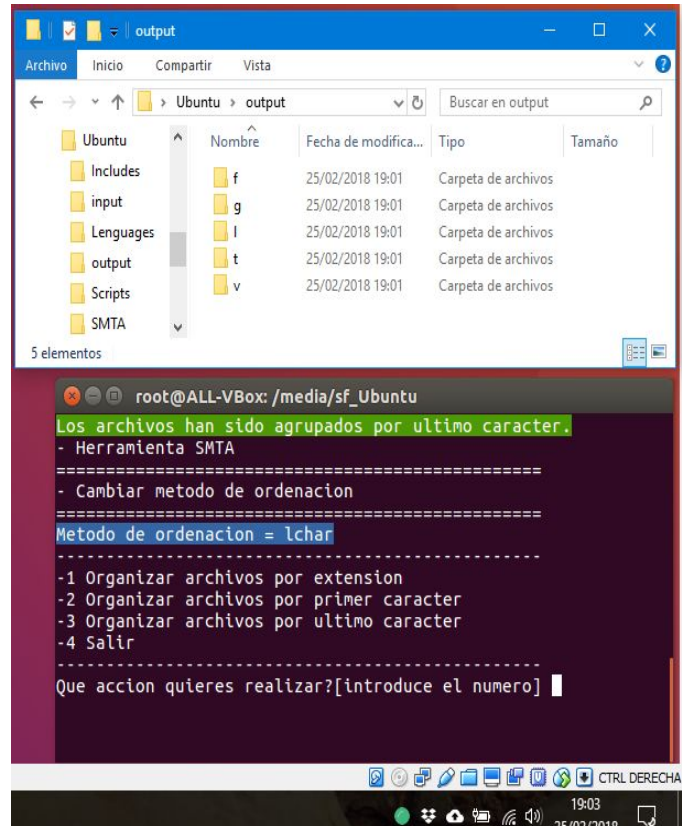
```
root@ALL-VBox: /media/sf_Ubuntu
- Revisar los archivos ordenados
- Legenda de los colores
=====
Archivos correctos
Archivos que no han sido localizados
=====
/media/sf_Ubuntu/output/A/Andreu.txt
/media/sf_Ubuntu/output/A/Animal.pdf
/media/sf_Ubuntu/output/c/config
/media/sf_Ubuntu/output/F/Felix.org.txt
/media/sf_Ubuntu/output/i/indice.html
/media/sf_Ubuntu/output/L/Local.kl.html
/media/sf_Ubuntu/output/P/Pedro.csv
/media/sf_Ubuntu/output/P/Powerpoint.arg.pdf
/media/sf_Ubuntu/output/P/Presentacion.oid.txt
/media/sf_Ubuntu/output/t/txt.jorge.papel.txt
/media/sf_Ubuntu/output/t/txt.pablo.guerrero.txt
=====
-1 Mirar por archivos extraviados en el directorio de entrada
-2 Eliminar los archivos extraviados
-3 Volver atras
=====
Que accion quieres realizar?[introduce el numero]
```

```
root@ALL-VBox: /media/sf_Ubuntu
Ya se esta utilizando este tipo de agrupacion.
- Herramienta SMTA
- Cambiar metodo de ordenacion
=====
Metodo de ordenacion = fchar
=====
-1 Organizar archivos por extension
-2 Organizar archivos por primer caracter
-3 Organizar archivos por ultimo caracter
-4 Salir
=====
Que accion quieres realizar?[introduce el numero]
```



Se prueba de hacer ahora la ordenación por extensión, entonces en la máquina anfitriona se puede ver desde el directorio de salida que los a agrupado por extensión y si se hace memoria había un archivo que no tenía extensión entonces crea un directorio llamado Unknown y lo mueve allí dentro.

Como última prueba de agrupación de los archivos por último carácter se puede apreciar que coge ese dato, lo transforma en un directorio y lo mete dentro de este.



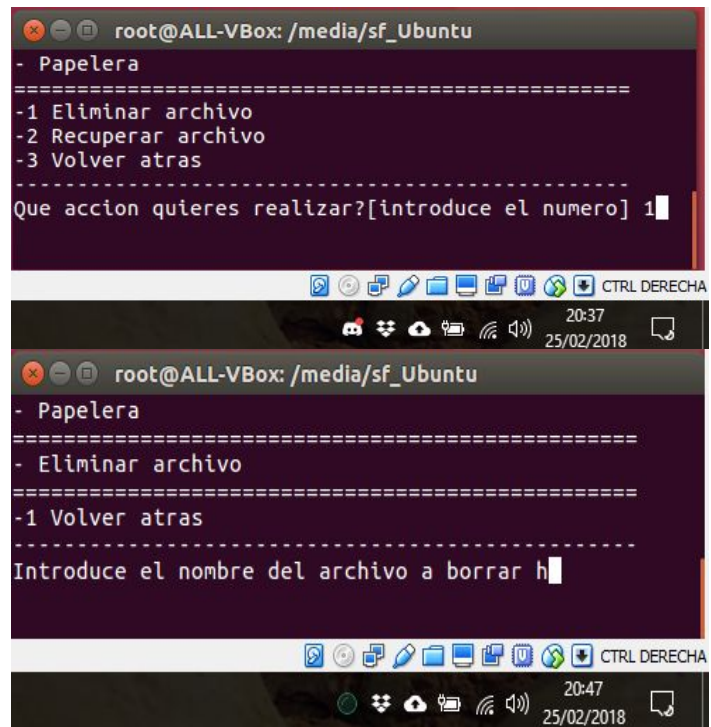
#### - Papelera de reciclaje

Para acceder a ella desde el primer menú del programa, se presiona en 2 en gestionar la papelera.

En esta hay dos acciones eliminar y recuperar archivos.

La papelera funciona de manera que si tu quieres algún archivo para enviarla a ella este tiene que coincidir con lo que pongas como búsqueda en el caso que busques un archivo que contenga la letra h.

Entonces aparecerá un listado por esta búsqueda y te dará dos opciones, que le indique cual archivo borrar





solo presionando el número correspondiente a este o bien volver atrás pulsando enter.

Más adelante te preguntará si estas seguro de borrarlo y marcas y o n como opción. Por último faltaría comprobar que se a borrando en la base de datos mostrada anteriormente o también haciendo una búsqueda otra vez en la papelera por parámetros introducidos.

Si se quiere recuperar este archivo habría que acceder al menú de papelera allí marca la opción 2 recuperar archivos.

```
root@ALL-VBox: /media/sf_Ubuntu
- Papelera
- Eliminar archivo
=====
-1 indice.html
-2 Local.kl.html
-3 0-xarchivex.txt
=====
Para volver atras pulsa enter en blanco
Introduce el nombre del archivo a borrar
```

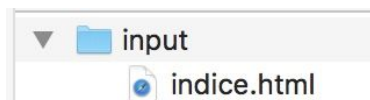
```
root@ALL-VBox: /media/sf_Ubuntu
Archivo eliminado
- Papelera
- Eliminar archivo
=====
-1 Volver atras
=====
Introduce el nombre del archivo a borrar
```

Funciona de la misma manera que a la hora de borrar un archivo, se tendría que buscar por la letra **h**, presionar en ( **y o n** ) y recuperarías ese archivo otra vez.

Sin embargo, si se cambió el método de ordenación entonces si el directorio donde se encontraba cuando se eliminó no existe ya, se avisará al usuario de que este ya no se encuentra y si quiere recuperar el archivo al directorio de entrada.

```
all@ALL-VBox: ~
No se ha podido restaurar al directorio anterior.
Deseas moverlo al directorio de entrada? [y/n]
```

Entonces al aceptar este se recuperará en el directorio de entrada.

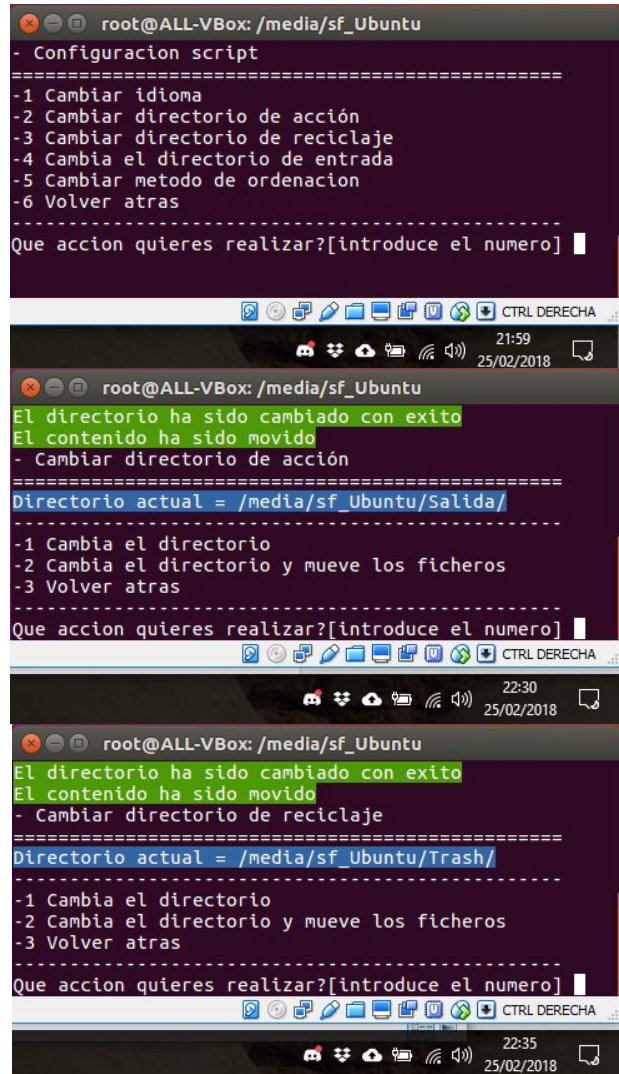


- Configuración:

1. **Cambiar idioma:** Permite seleccionar entre dos idiomas (Español o Ingles).
2. **Cambiar directorio de acción:** Permite hacer dos opciones bien cambiar de directorio de acción o bien cambiar de directorio de acción (te preguntará la ruta absoluta de este nuevo directorio, recuerda indicarle cuando acaba el directorio con /) y mover los archivos al nuevo directorio que se les asigne como de acción.
3. **Cambiar directorio de reciclaje:**

Es exactamente lo mismo que el apartado mencionado anteriormente, tiene las mismas opciones, para cambiar de directorio y mover archivos dentro de él hay que poner la ruta absoluta con su nuevo nombre y cerrandola con /.

4. **Cambiar directorio de entrada:** Es exactamente la misma funcionalidad y con las mismas opciones que los apartados anteriores de configuración.
5. **Cambiar método de ordenación:** Este apartado está ya probado anteriormente en las pruebas realizadas.



The image shows three sequential screenshots of a terminal window running a configuration script. The terminal title is 'root@ALL-VBox: /media/sf\_Ubuntu'. The first screenshot shows the 'Configuracion script' menu with options 1-6. The second screenshot shows the result of option 2: 'El directorio ha sido cambiado con exito' and 'El contenido ha sido movido', followed by a new menu for changing the action directory to '/media/sf\_Ubuntu/Salida/'. The third screenshot shows the result of option 3: 'El directorio ha sido cambiado con exito' and 'El contenido ha sido movido', followed by a new menu for changing the recycle bin directory to '/media/sf\_Ubuntu/Trash/'.

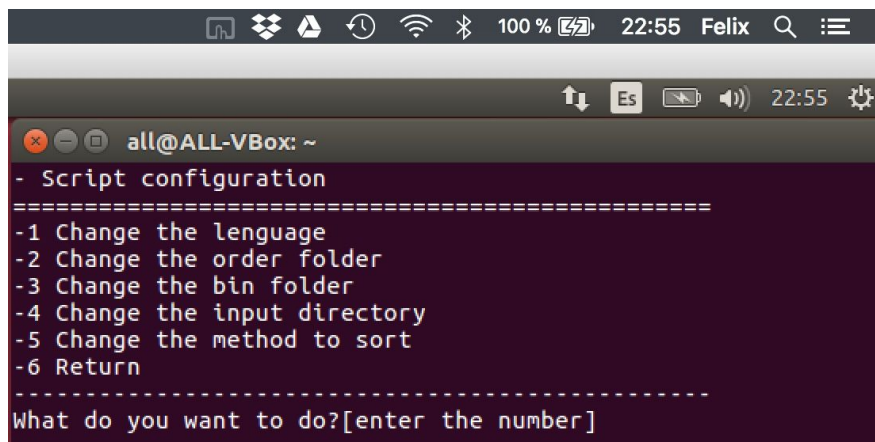
```
root@ALL-VBox: /media/sf_Ubuntu
- Configuracion script
=====
-1 Cambiar idioma
-2 Cambiar directorio de acción
-3 Cambiar directorio de reciclaje
-4 Cambia el directorio de entrada
-5 Cambiar metodo de ordenacion
-6 Volver atras
=====
Que accion quieres realizar?[introduce el numero]

root@ALL-VBox: /media/sf_Ubuntu
El directorio ha sido cambiado con exito
El contenido ha sido movido
- Cambiar directorio de acción
=====
Directorio actual = /media/sf_Ubuntu/Salida/
=====
-1 Cambia el directorio
-2 Cambia el directorio y mueve los ficheros
-3 Volver atras
=====
Que accion quieres realizar?[introduce el numero]

root@ALL-VBox: /media/sf_Ubuntu
El directorio ha sido cambiado con exito
El contenido ha sido movido
- Cambiar directorio de reciclaje
=====
Directorio actual = /media/sf_Ubuntu/Trash/
=====
-1 Cambia el directorio
-2 Cambia el directorio y mueve los ficheros
-3 Volver atras
=====
Que accion quieres realizar?[introduce el numero]
```

- Sistema de lenguaje:

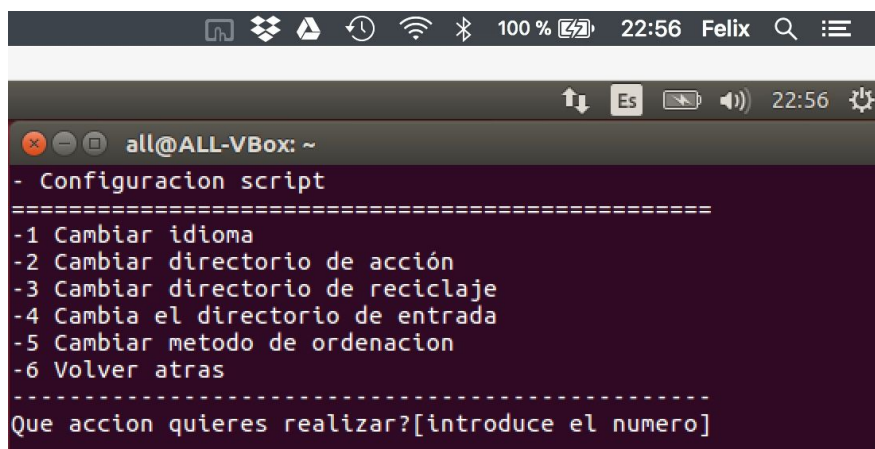
El sistema al iniciar la herramienta obtiene el campo que indica que lenguaje es, al ser el mismo que el nombre del fichero logra la dirección del fichero que contiene todos los strings del idioma al cual hace referencia. Entonces para conseguir el string utiliza una función que va a buscar el identificativo correcto en la base de idiomas, cada línea tiene que estar identificada por la letra “s” más un número ordenado desde la primera línea del archivo hasta que se acaben las acciones en los respectivos archivos que forman el sistema de lenguaje.



```
all@ALL-VBox: ~  
- Script configuration  
=====
```

-1	Change the language
-2	Change the order folder
-3	Change the bin folder
-4	Change the input directory
-5	Change the method to sort
-6	Return

```
-----  
What do you want to do?[enter the number]
```



```
all@ALL-VBox: ~  
- Configuracion script  
=====
```

-1	Cambiar idioma
-2	Cambiar directorio de acción
-3	Cambiar directorio de reciclaje
-4	Cambia el directorio de entrada
-5	Cambiar metodo de ordenacion
-6	Volver atras

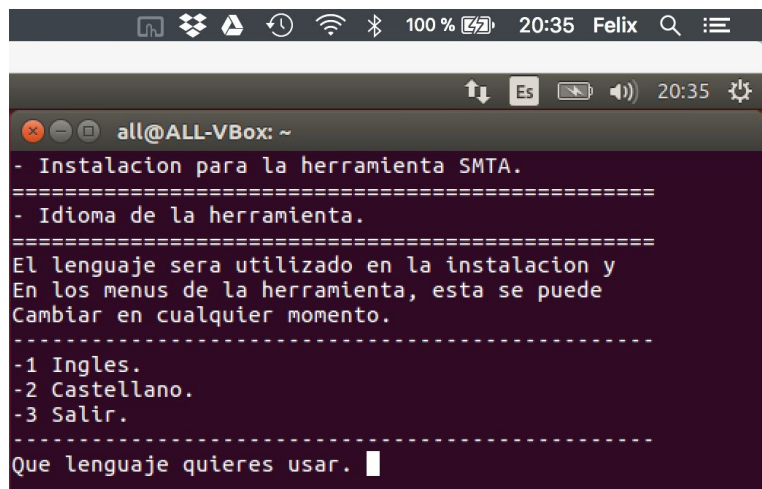
```
-----  
Que accion quieres realizar?[introduce el numero]
```

## - Instalador:

Para iniciar la instalación, se ejecuta el archivo SMTA/installer.sh se selecciona el idioma para el instalador y se ejecuta la acción de instalación.

Ahora el usuario realizará una serie de pasos, los cuales son utilizados para configurar la herramienta al final del proceso.

El primer paso es seleccionar el idioma de la herramienta. aquí se puede seleccionar entre los varios idiomas que ofrece la herramienta



```
all@ALL-VBox: ~  
- Instalacion para la herramienta SMTA.  
=====
```

- Idioma de la herramienta.  
=====

El lenguaje sera utilizado en la instalacion y  
En los menus de la herramienta, esta se puede  
Cambiar en cualquier momento.  
=====

-1 Ingles.  
-2 Castellano.  
-3 Salir.  
=====

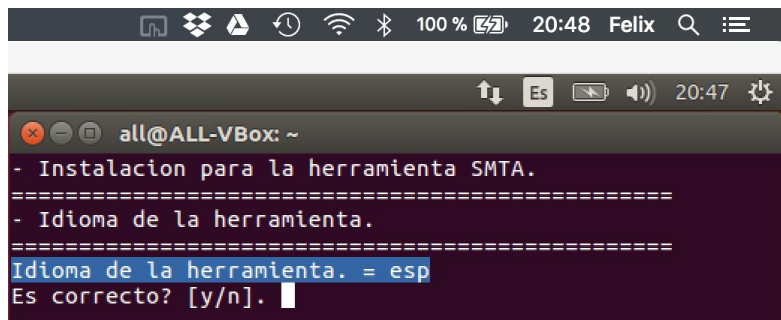
Que lenguaje quieres usar. █

A continuación aparecerá un mensaje para confirmar el idioma seleccionado.

Esta ventana se utiliza para confirmar los datos introducidos y irá apareciendo después de cada paso realizado por el usuario para

asegurar que la instalación se realiza como el usuario quiere. Si el usuario pulsara “n”, entonces volvería a la configuración del parámetro. si pulsara “y” pasaría al siguiente paso.

El siguiente paso sería introducir la ruta del directorio donde se quiere instalar la herramienta, hay que tener en cuenta que los directorios no los creará la herramienta de instalación así que estos tienen que hacerse antes de la ejecución del instalador. Una vez puesta la ruta, este comprueba que exista y como en el paso anterior, el usuario tiene que confirmar que la ruta impuesta es correcta.



```
all@ALL-VBox: ~  
- Instalacion para la herramienta SMTA.  
=====
```

- Idioma de la herramienta.  
=====

Idioma de la herramienta. = esp  
Es correcto? [y/n]. █



```
all@ALL-VBox: ~  
Entrada introducida.  
- Instalacion para la herramienta SMTA.  
=====
```

- Ruta de la instalacion.  
=====

La ruta de instalación indica donde se instalara y donde estaran los archivos utilizados por la herramienta para poder funcionar.  
=====

-1 Volver.  
=====

Introduce una ruta absoluta. █

A partir de aquí, los otros pasos a realizar son iguales a los dos explicados. Estos son el método de ordenación, el directorio de entrada de ficheros, el directorio de salida y el directorio para la papelera.

Entonces al finalizar de introducir los parámetros necesarios para la instalación de la herramienta, aparecerá un menú donde mostrarán toda la información que introdujo el usuario para realizar una última inspección antes de instalar la herramienta.

Al ejecutar la instalación se mostrará por pantalla como se descomprime los archivos y mostrará el mensaje de finalización de la instalación en

```
all@ALL-VBox: ~  
Entrada introducida.  
- Instalacion para la herramienta SMTA.  
=====
```

- Revisa los parametros introducidos.  
=====

Idioma de la herramienta. = esp  
Ruta de la instalacion. = /media/sf\_Ubuntu/inta/  
Metodo de ordenación. = ext  
Ruta para el directorio de entrada. = /media/sf\_Ubuntu/i  
nput/  
Ruta para el directorio de salida. = /media/sf\_Ubuntu/ou  
t/  
Ruta para la papelera. = /media/sf\_Ubuntu/trash/  
=====

Instalacion, [y/n]. █

```
all@ALL-VBox: ~  
La instalación esta en curso.  
Archive: /media/sf_Ubuntu/SMTA/Includes/Archive.zip  
creating: /media/sf_Ubuntu/inta/Includes/  
creating: /media/sf_Ubuntu/inta/Includes/Data/  
creating: /media/sf_Ubuntu/inta/Includes/Lib/  
inflating: /media/sf_Ubuntu/inta/Includes/Lib/control.sh  
inflating: /media/sf_Ubuntu/inta/Includes/Lib/menu.sh  
inflating: /media/sf_Ubuntu/inta/Includes/Lib/order.sh  
creating: /media/sf_Ubuntu/inta/Languages/  
inflating: /media/sf_Ubuntu/inta/Languages/eng.txt  
inflating: /media/sf_Ubuntu/inta/Languages/esp.txt  
creating: /media/sf_Ubuntu/inta/Scripts/  
inflating: /media/sf_Ubuntu/inta/Scripts/change.sh  
inflating: /media/sf_Ubuntu/inta/Scripts/check.sh  
inflating: /media/sf_Ubuntu/inta/Scripts/config.sh  
inflating: /media/sf_Ubuntu/inta/Scripts/mor.sh  
inflating: /media/sf_Ubuntu/inta/Scripts/smta.sh  
inflating: /media/sf_Ubuntu/inta/Scripts/sort.sh  
inflating: /media/sf_Ubuntu/inta/Scripts/trash.sh  
La instalación esta en curso.  
=====
```

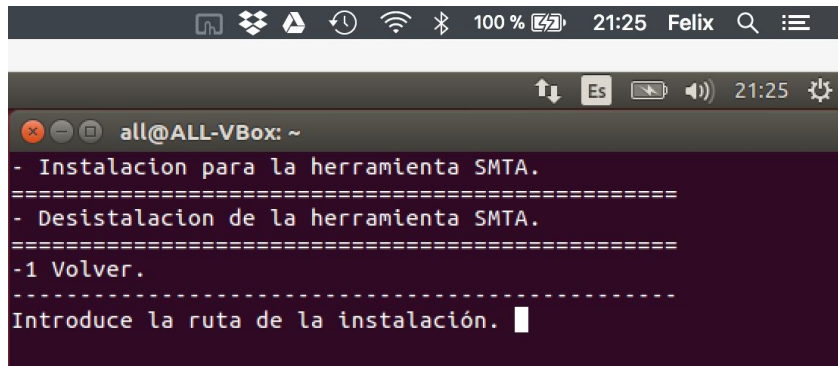
verde por lo que indicaría que fue correcta la instalación.

Finalmente volvemos al menú principal de la herramienta de instalación donde saldríamos para finalizar la instalación.



## - Desinstalador:

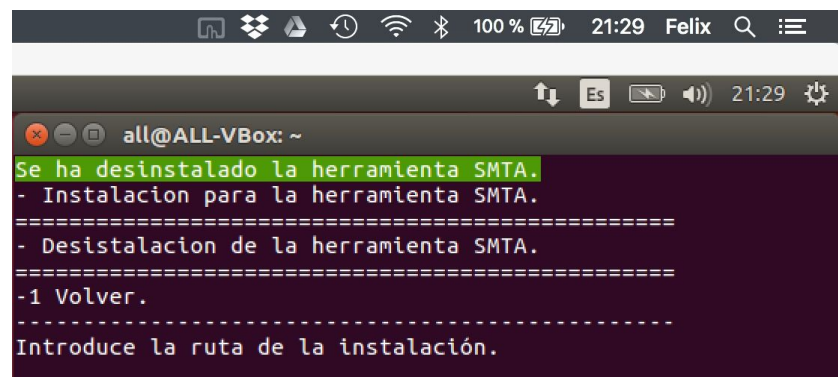
Para desinstalar la herramienta se accede a la misma herramienta utilizada para ejecutar la instalación pero esta vez se realizaria la acción de desinstalar, esta nos lleva al siguiente menú:



```
all@ALL-VBox: ~  
- Instalacion para la herramienta SMTA.  
=====
```

The screenshot shows a terminal window titled 'all@ALL-VBox: ~'. It displays a menu with the following options: '- Instalacion para la herramienta SMTA.', '- Desinstalacion de la herramienta SMTA.', and '-1 Volver.'. The menu is separated by lines of equals signs. At the bottom, it prompts 'Introduce la ruta de la instalación.' with a cursor.

Entonces se tendrá que introducir la ruta de la instalación, este tiene un seguro para evitar que se borren directorios principales ya que busca archivos que pertenecen a la herramienta en el directorio para evitar problemas con directorios del sistema.



```
all@ALL-VBox: ~  
Se ha desinstalado la herramienta SMTA.  
- Instalacion para la herramienta SMTA.  
=====
```

The screenshot shows the same terminal window as before, but now with a green highlight on the message 'Se ha desinstalado la herramienta SMTA.' at the top. The menu options and the prompt 'Introduce la ruta de la instalación.' are still visible below.

Una vez ejecutada, en la consola de instalación aparecerá el mensaje de que se ha ejecutado correctamente la operación.

Se recomienda realizar la instalación en un directorio limpio para que al desinstalar no se eliminen archivos que no formen parte de la herramienta.

---

## Organización del proyecto:

El reparto que se ha realizado al final ha sido entre dos persona siendo de esta forma:

Formante	Módulos	Nota a aspirar
Félix Aguilar Ferrer	Instalador, desinstalador, menu, tratado de base de datos, papelera y la configuración exceptuado el método de ordenación.	Mis primeras expectativas eran un 9 pero a lo largo del desarrollo de la herramienta creo que me merezco un 10.
Andreu Llabres Bañuls	Métodos de ordenación (por extensión, primer carácter y último carácter) y por último configuración de estos.	Mis primeras expectativas eran un 5 pero a lo largo del desarrollo del programa creo que me merezco un 7.

## Puntos de discusión:

### Elección del proyecto y división:

El proyecto se decidió de forma unánime por los cuatro miembros del grupo inicial entre varias opciones obtenidas después de una lluvia de ideas, finalmente se decidió realizar una herramienta de gestión de archivos, ya que permitía un diseño modular y terreno donde experimentar además de poder realizar suficiente para las notas que aspiraban los miembros de grupo.

### Abandono de miembros del grupo:

Cuando uno de los miembros abandonó el grupo al ver que no llegaba al nivel requerido, este dejó su parte libre, esta fue dada al otro miembros de este, el cual más adelante también abandonó por motivos similares. Entonces para que el proyecto no quedara sin sistema de papelera esta parte se dio a Felix para que hiciera un sistema suficiente para este módulo.

---

### Propuestas del profesor:

#### **Andreu Llabres Bañuls:**

La recomendación que hizo el profesor en el método de ordenación por extensión, fue que creara una función que contase cuantos puntos contenía el archivo que tratase para así saber coger el último de ellos, la función es **countp** ubicada dentro de **order.sh**.

#### **Felix Aguilar Ferrer:**

La recomendación del profesor que me hizo, fue que en la función **divider**, se le permitiera cambiar el caracter para dividir los apartados a la vez que se puede modificar la longitud de la línea divisora.

### **Problemas encontrados:**

#### Abandono de miembros del grupo:

Durante el desarrollo del proyecto dos miembros del grupo optaron por ir a examen ya que vieron que el trabajo sobrepasaba sus conocimientos que tenían de Linux.

Esto no se solucionó, pero entre los restantes miembros se pudo empujar el proyecto y completarlo, dejando un poco de funcionalidad que iba a estar presente en la versión final de lado.

#### Borrado masivo del sistema:

Cuando estábamos juntando el proyecto la parte de Andreu que es la ordenación de archivos con una función para resetear los archivos al directorio de entrada hecho por Félix, el problema que hubo fue que la máquina virtual en la cual Andreu estaba trabajando en el proyecto, provo a ver si funcionaba dicho conjunto y lo que hizo fue borrar el sistema entero haciendo un rm de cada directorio y subdirectorio con su contenido.

La solución fue modificar la parte de ordenación por último caracter ya que cuando hacia un mv al directorio con el archivo este movia espacios en blanco y aparte la función **resetfiles** los parámetros que necesitaba que eran las rutas en las cuales debía trabajar no estaban bien definidas.

---

## Bibliografía:

1. <http://tldp.org/LDP/abs/html/string-manipulation.html>
2. <http://www.compciv.org/topics/bash/loops/>
3. Apuntes de clase.
4. StackOverflow.