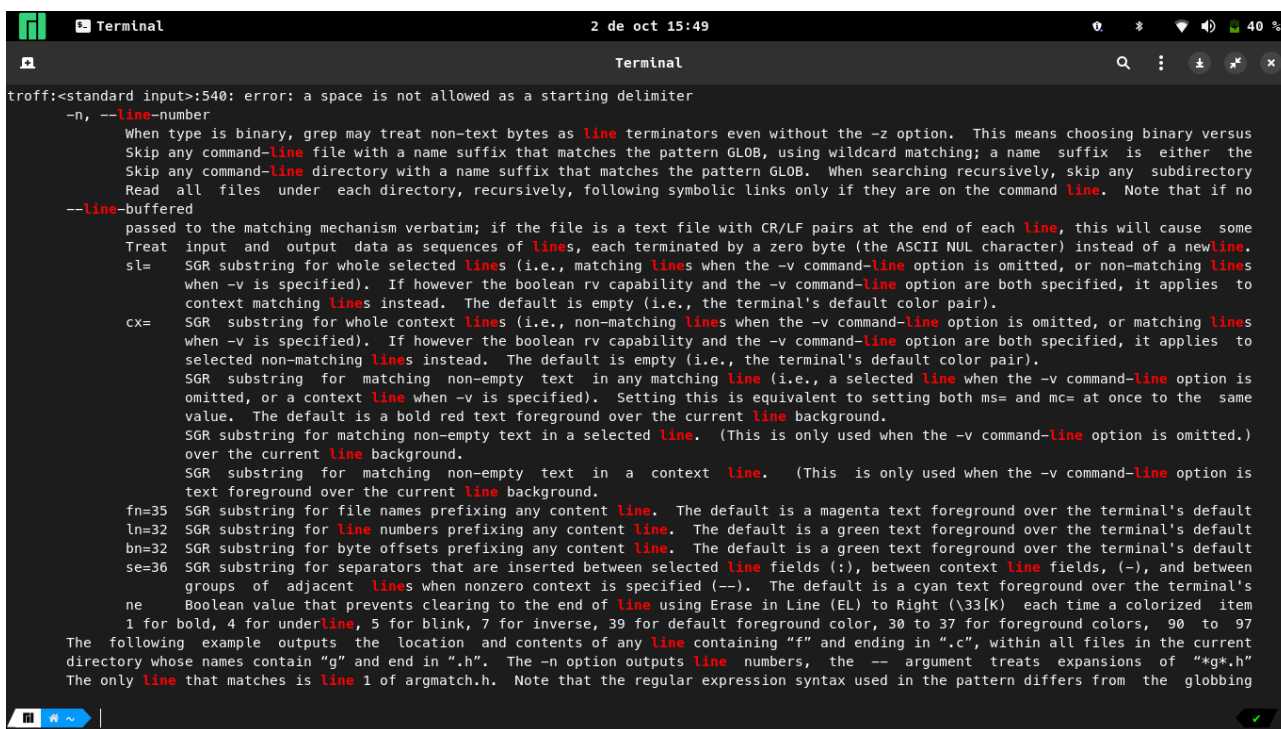


Ejercicio de entorno SHELL

1.- Realiza las siguientes operaciones:

- Consulta los manuales del comando **grep** y muestra todas las líneas en las que aparece la cadena 'line'. Fíjate en la salida obtenida por pantalla.

man grep | **grep 'line'**




```
troff:<standard input>:540: error: a space is not allowed as a starting delimiter
-n, --line-number
    When type is binary, grep may treat non-text bytes as line terminators even without the -z option. This means choosing binary versus
    Skip any command-line file with a name suffix that matches the pattern GLOB, using wildcard matching; a name suffix is either the
    Skip any command-line directory with a name suffix that matches the pattern GLOB. When searching recursively, skip any subdirectory
    Read all files under each directory, recursively, following symbolic links only if they are on the command line. Note that if no
--line-buffered
    passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some
    Treat input and output data as sequences of lines, each terminated by a zero byte (the ASCII NUL character) instead of a new line.
    s= SGR substring for whole selected lines (i.e., matching lines when the -v command-line option is omitted, or non-matching lines
    when -v is specified). If however the boolean rv capability and the -v command-line option are both specified, it applies to
    context matching lines instead. The default is empty (i.e., the terminal's default color pair).
    cx= SGR substring for whole context lines (i.e., non-matching lines when the -v command-line option is omitted, or matching lines
    when -v is specified). If however the boolean rv capability and the -v command-line option are both specified, it applies to
    selected non-matching lines instead. The default is empty (i.e., the terminal's default color pair).
    SGR substring for matching non-empty text in any matching line (i.e., a selected line when the -v command-line option is
    omitted, or a context line when -v is specified). Setting this is equivalent to setting both ms= and mc= at once to the same
    value. The default is a bold red text foreground over the current line background.
    SGR substring for matching non-empty text in a selected line. (This is only used when the -v command-line option is omitted.)
    over the current line background.
    SGR substring for matching non-empty text in a context line. (This is only used when the -v command-line option is
    text foreground over the current line background.
    fn=35 SGR substring for file names prefixing any content line. The default is a magenta text foreground over the terminal's default
    ln=32 SGR substring for line numbers prefixing any content line. The default is a green text foreground over the terminal's default
    bn=32 SGR substring for byte offsets prefixing any content line. The default is a green text foreground over the terminal's default
    se=36 SGR substring for separators that are inserted between selected line fields (:), between context line fields, (-), and between
    groups of adjacent lines when nonzero context is specified (--). The default is a cyan text foreground over the terminal's
    ne Boolean value that prevents clearing to the end of line using Erase in Line (EL) to Right (\33[K) each time a colored item
    1 for bold, 4 for underline, 5 for blink, 7 for inverse, 39 for default foreground color, 30 to 37 for foreground colors, 90 to 97
    The following example outputs the location and contents of any line containing "f" and ending in ".c", within all files in the current
    directory whose names contain "g" and end in ".h". The -n option outputs line numbers, the -- argument treats expansions of "*g.h"
    The only line that matches is line 1 of argmatch.h. Note that the regular expression syntax used in the pattern differs from the globbing
```

Ejercicio de entorno SHELL

- Repite ahora la operación anterior redireccionando la salida estándar a un fichero. Comprueba el contenido del fichero de salida.

man grep | grep 'line' >fichero_de_salida.txt



```
Terminal
2 de oct 16:34

as infinity and grep does not stop; this is the default. If the input is standard input from a regular file, and NUM matching lines
are output, grep ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the
presence of trailing context lines. This enables a calling process to resume a search. When grep stops after NUM matching lines, it
outputs any trailing context lines. When the -c or --count option is also used, grep does not output a count greater than NUM. When
the -v or --invert-match option is also used, grep stops after outputting NUM non-matching lines.

-n, --line-number
When type is binary, grep may treat non-text bytes as line terminators even without the -z option. This means choosing binary versus
Skip any command-line file with a name suffix that matches the pattern GLOB, using wildcard matching; a name suffix is either the
Skip any command-line directory with a name suffix that matches the pattern GLOB. When searching recursively, skip any subdirectory
Read all files under each directory, recursively, following symbolic links only if they are on the command line. Note that if no
--line-buffered
passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some
Treat input and output data as sequences of lines, each terminated by a zero byte (the ASCII NUL character) instead of a newline.
sl= SGR substring for whole selected lines (i.e., matching lines when the -v command-line option is omitted, or non-matching lines
when -v is specified). If however the boolean rv capability and the -v command-line option are both specified, it applies to
context matching lines instead. The default is empty (i.e., the terminal's default color pair).
cx= SGR substring for whole context lines (i.e., non-matching lines when the -v command-line option is omitted, or matching lines
when -v is specified). If however the boolean rv capability and the -v command-line option are both specified, it applies to
selected non-matching lines instead. The default is empty (i.e., the terminal's default color pair).
SGR substring for matching non-empty text in any matching line (i.e., a selected line when the -v command-line option is
omitted, or a context line when -v is specified). Setting this is equivalent to setting both ms= and mc= at once to the same
value. The default is a bold red text foreground over the current line background.
SGR substring for matching non-empty text in a selected line. (This is only used when the -v command-line option is omitted.)
over the current line background.
SGR substring for matching non-empty text in a context line. (This is only used when the -v command-line option is
text foreground over the current line background.
fn=35 SGR substring for file names prefixing any content line. The default is a magenta text foreground over the terminal's default
ln=32 SGR substring for line numbers prefixing any content line. The default is a green text foreground over the terminal's default
bn=32 SGR substring for byte offsets prefixing any content line. The default is a green text foreground over the terminal's default
se=36 SGR substring for separators that are inserted between selected line fields (:), between context line fields, (-), and between
groups of adjacent lines when nonzero context is specified (--). The default is a cyan text foreground over the terminal's
ne Boolean value that prevents clearing to the end of line using Erase in Line (EL) to Right (\33[K) each time a colorized item
1 for bold, 4 for underline, 5 for blink, 7 for inverse, 39 for default foreground color, 30 to 37 for foreground colors, 90 to 97
The following example outputs the location and contents of any line containing "f" and ending in ".c", within all files in the current
```

- Intenta copiar fichero_de_salida sobre sí mismo. Fíjate en la salida obtenida por pantalla.

cp fichero_de_salida.txt fichero_de_salida.txt



```
Terminal
2 de oct 16:36

> cp fichero_de_salida.txt fichero_de_salida.txt
cp: 'fichero_de_salida.txt' y 'fichero_de_salida.txt' son el mismo fichero
```

Ejercicio de entorno SHELL

- Repite ahora la operación anterior redireccionando el error estándar a un fichero. Comprueba el contenido del fichero de error.

cp fichero_de_salida.txt fichero_de_salida.txt 2> fichero_de_error.txt



```
Terminal
2 de oct 16:40
cp: 'fichero_de_salida.txt' y 'fichero_de_salida.txt' son el mismo fichero
cp: 'fichero_de_salida.txt' y 'fichero_de_salida.txt' 2> fichero_de_error.txt
cat fichero_de_error.txt
File: fichero_de_error.txt
1 cp: 'fichero_de_salida.txt' y 'fichero_de_salida.txt' son el mismo fichero
```

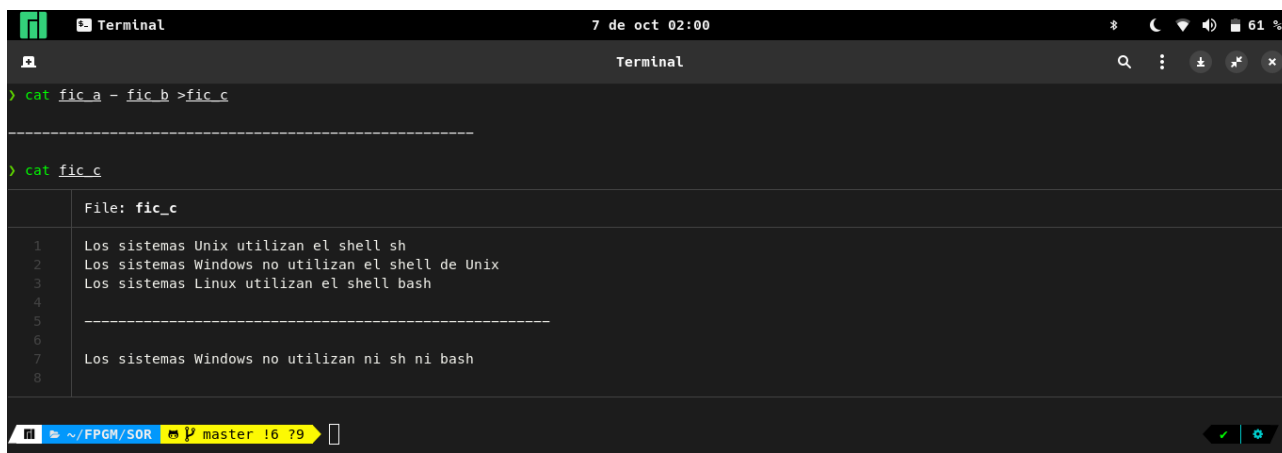
2.- Realiza las siguientes operaciones:

- Crea mediante un único comando un fichero que muestre dos ficheros cualesquiera separándolos por:

- una líneas en blanco
- una línea que contenga un literal cualquiera
- otra línea en blanco.

Utiliza para ello el comando **cat** con la opción **'-'** para introducir el texto de separación entre los dos ficheros por la **entrada estándar** (teclado).

cat fic_a - fic_b >fic_c




```
Terminal
7 de oct 02:00
cat fic_a - fic_b >fic_c
cat fic_c
File: fic_c
1 Los sistemas Unix utilizan el shell sh
2 Los sistemas Windows no utilizan el shell de Unix
3 Los sistemas Linux utilizan el shell bash
4 -----
5
6 Los sistemas Windows no utilizan ni sh ni bash
7
8
```

Ejercicio de entorno SHELL

- Repite a continuación esta operación redireccionando la **entrada estándar** del proceso desde un fichero. Deberás crear para ello un fichero que contenga:

- Una línea con la cadena "-----"
- una línea que contenga un literal cualquiera
- Una línea con la cadena "-----"

```
 Terminal 7 de oct 02:08 * 🌙 🔊 🔋 57 %  
 Terminal 🔍 ⋮ 👤 ✖️ ×  
> cat fic_a fic_c fic_b >result_fic_acb  
> cat result_fic_ach  
File: result_fic_acb  
1 Los sistemas Unix utilizan el shell sh  
2 Los sistemas Windows no utilizan el shell de Unix  
3 Los sistemas Linux utilizan el shell bash  
4 -----  
5 kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk  
6 -----  
7 Los sistemas Windows no utilizan ni sh ni bash  
8
```

3.- Crea 2 ficheros con un editor de textos:

fic_a :

Los sistemas Unix utilizan el shell sh
Los sistemas Windows no utilizan el shell de Unix
Los sistemas Linux utilizan el shell bash


fic_b:

Los sistemas Windows no utilizan ni sh ni bash

Ejercicio de entorno SHELL

- Cuenta el número total de líneas (entre ambos ficheros) en las que aparece la palabra 'bash'.

`cat fic_a fic_b | grep -wc bash`



```
Terminal 7 de oct 02:40
kike@kike:~/FPGM/SOR
> cat fic_a fic_b | grep -wc bash
2
```

- Cuenta el número total de líneas en las que aparecen las palabras 'sh' y 'bash'.

`cat fic_a fic_b | grep -Ewc 'sh|bash'`



```
Terminal 7 de oct 02:36
kike@kike:~/FPGM/SOR
> cat fic_a fic_b | grep -Ewc 'sh|bash'
Los sistemas Unix utilizan el shell sh
Los sistemas Linux utilizan el shell bash
Los sistemas Windows no utilizan ni sh ni bash
> cat fic_a fic_b | grep -Ewc 'sh|bash'
3
```

4.- ¿Qué orden ejecutarías para que te saliese por pantalla?:
Soy el usuario `tu_identificativo_de_usuario` y mi directorio
`$HOME` es `tu_dir_home`?

`echo "Soy el usuario $USER y mi directorio $HOME es $HOME"`



```
Terminal 2 de oct 16:50
Terminal
> echo "Soy el usuario $USER y mi directorio $HOME es $HOME"
Soy el usuario kike y mi directorio $HOME es /home/kike
```

-`tu_identificativo_de_usuario`, es un valor que contiene una variable de entorno.

- `tu_dir_home`, es el contenido de la variable de entorno `$HOME`.