

# Bash

---

## Comandos **ls**

```
ls # Muestra los archivos no ocultos de un directorio.  
ls -a # Muestra todos los archivos, inclusive los ocultos.  
ls -l # Muestra los datos del archivo (tipo, cantidad de carpetas, fecha de  
creación, nombre).  
ls -la # Funciona igual que el "ls -l", pero muestra las carpetas anteriores a  
esta (con . o ..).
```

## Comandos **cd**

```
cd directorio # Cambiar de directorio  
cd # Cambia al directorio inicial.  
cd .. # Retrocede al directorio anterior.
```

## Comandos **rm**

```
rm * # Borra todo lo que se encuentre en el directorio en el que estes situado.  
rm *.extension # Borra todo lo que termine en alguna extension (.py, .c, .cpp,  
.css, .html, etc).
```

## Comandos **cat**

```
cat archivo # Mostrar contenido del archivo.
```

## Comandos **diff**

```
diff archivo1 archivo2 # Comparar el contenido del archivo.  
diff -y archivo1 archivo2 # Muestra las diferencias entre dos archivos linea a  
linea.
```

## Comandos **grep** (funciona con cualquier archivo de texto).

```
grep "Nombre" archivo # Buscar en el contenido del archivo una palabra en  
específico.  
grep "^Nombre" archivo # Busca la las lineas del archivo que empiezan por esa  
palabra.  
grep "Nombre$" archivo # Busca la las lineas del archivo que terminan por esa
```

```
palabra.  
grep "^$" archivo # Busca las líneas vacías.
```

Comandos **find** (funciona con cualquier archivo de texto).

```
find . -type d #Busca recursivamente en el directorio en el que estes los  
subdirectorios  
  
Añadido --help  
  
```bash  
comando --help # la ayuda del comando
```

Comando **man**

```
man comando # Manual de un comando.
```

Comando **head**

```
head archivo # Te muestra el principio de un archivo.  
head -n 2 archivo # Te muestra las dos primeras líneas de un archivo (el número  
puede variar dependiendo de las líneas que quieras seleccionar).
```

Comando **tail**

```
tail archivo # Te muestra el final de un archivo.  
tail -n 19 # Muestra las 19 últimas líneas.
```

Comando **less**

```
less archivo # Te permite irte moviendo por el archivo si es muy grande.
```

Enlazar comandos **PIPES**

```
cat archivo | grep "Nombre$" # Ejemplo de concatenación de comandos.  
grep "error" file1.txt | wc --word < archivo # Concatenar y crear archivo con el  
contenido de la salida de los comandos.
```

Comandos **sed**

```
sed 's/Palabra a remplazar/Reemplazo/g' archivo # Reemplaza una palabra por otra de un archivo.  
sed '/^$/d' archivo # Borra las líneas vacías  
sed '/^$/d' | uniq archivo # Borra dos líneas seguidas que estén vacías  
sed '/^$/d' archivo | sort | uniq # Ordena y borra las líneas vacías que estén seguidas  
sed '5,95d' archivo # Borra las líneas que hay entre las líneas 5 y 95 del archivo.
```

### Comandos **wc**

```
wc -w archivo # Cuenta la cantidad de palabras que hay en un archivo.  
wc -l archivo # Cuenta la cantidad de líneas.
```

### Comando **sort**

```
sort archivo # Ordena alfabéticamente las líneas del archivo.  
sort -b archivo # Cuenta las líneas de un archivo, ignorando los espacios en blanco
```

### Comando **uniq**

```
uniq archivo # Borra las líneas duplicadas del archivo.  
uniq -D archivo # Muestra las líneas duplicadas.
```

### Comando **pwd**

```
pwd # Ruta entera de donde te encuentras.
```

### Comando **>**

```
archivo1 > archivo2 # Pasa la información del archivo1 al archivo2 (sobrescribe).  
archivo3 >> archivo2 # Añade la información del archivo3 al archivo2 (a continuación)
```

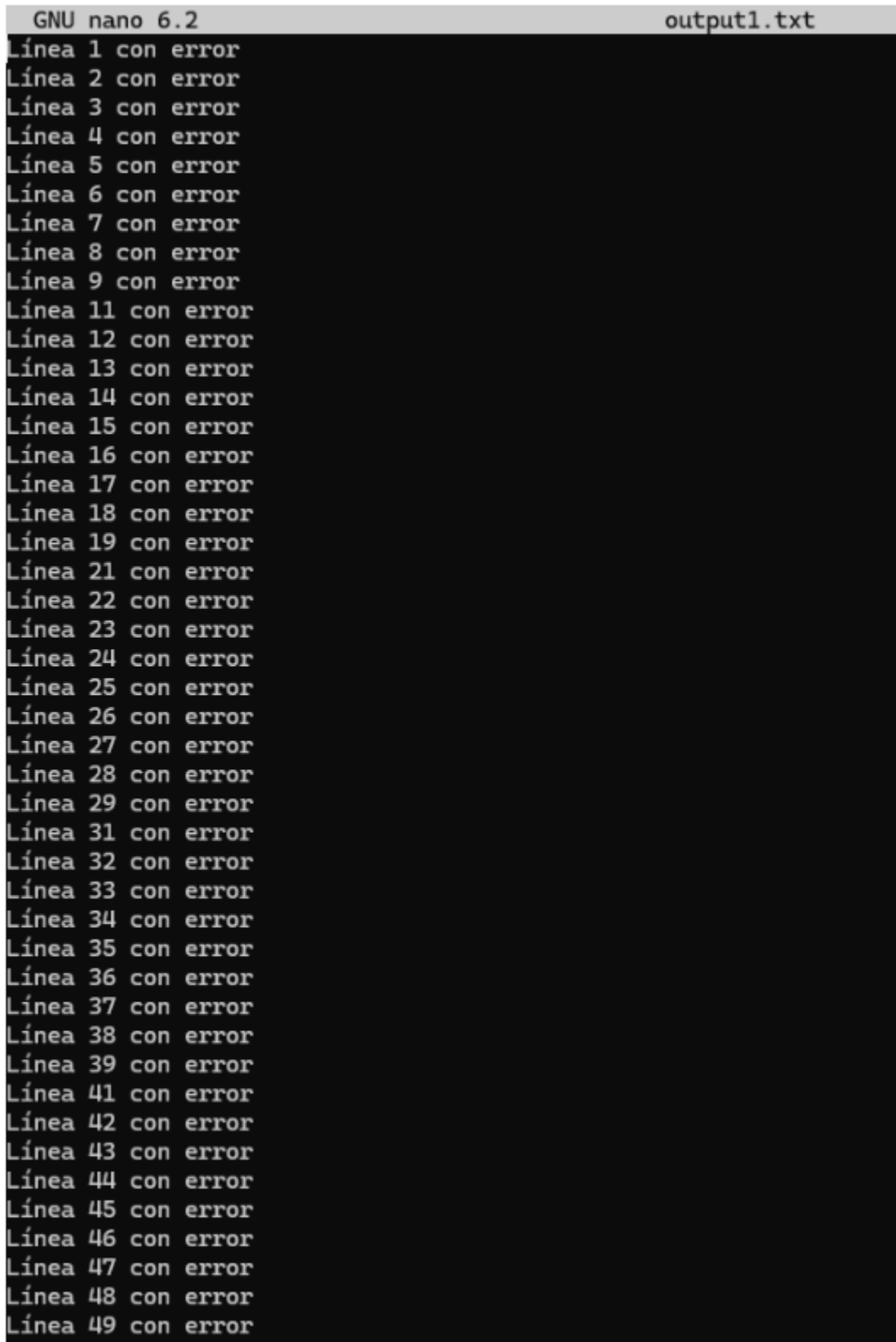
## Ejercicios

---

### Ejercicio 1

1. Encuentra todas las líneas que terminen con "error" en el archivo `file1.txt` y guarda el resultado en `output1.txt`.

```
grep "error" file1.txt > output1.txt
```

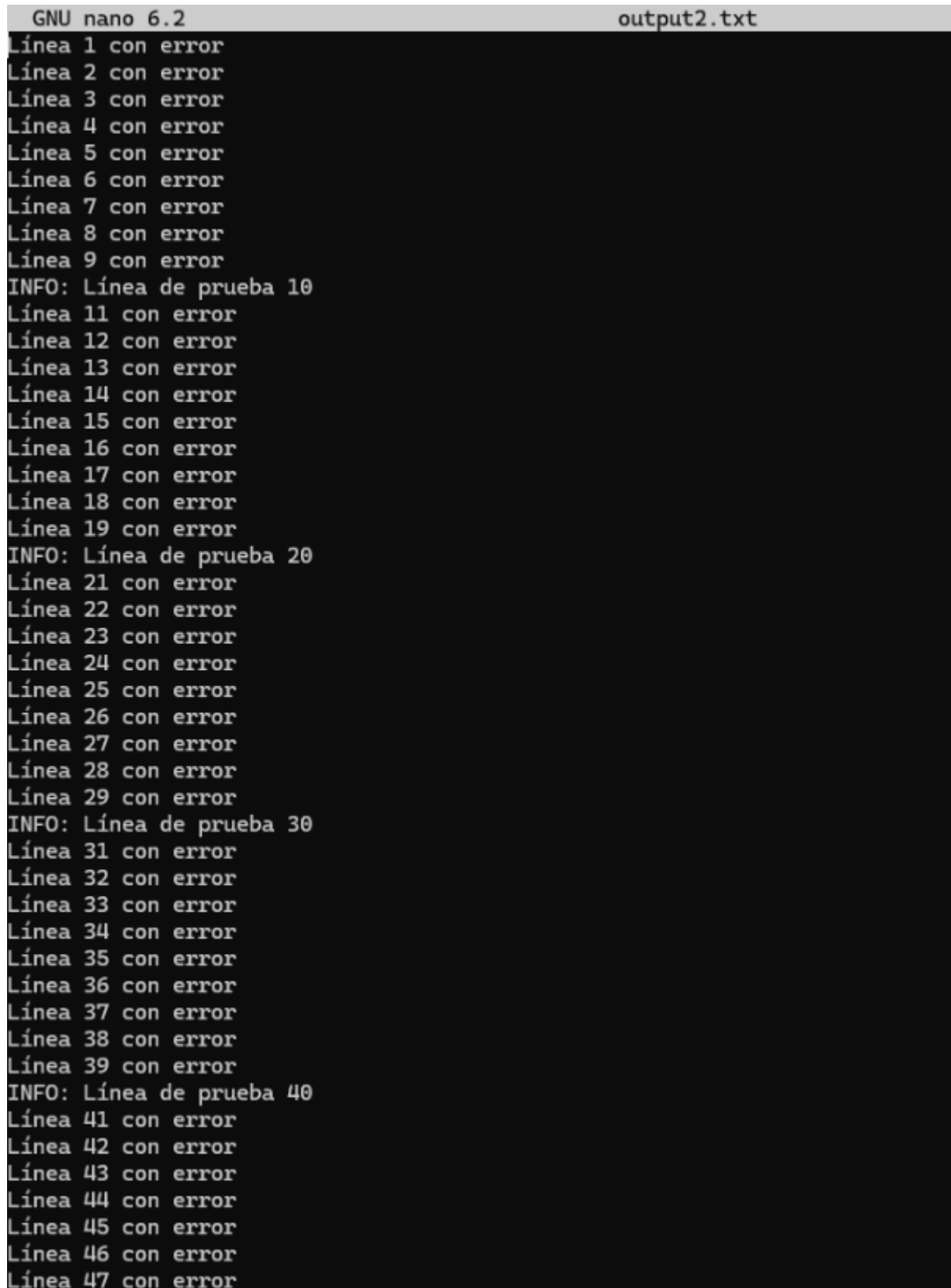


```
GNU nano 6.2 output1.txt
Línea 1 con error
Línea 2 con error
Línea 3 con error
Línea 4 con error
Línea 5 con error
Línea 6 con error
Línea 7 con error
Línea 8 con error
Línea 9 con error
Línea 11 con error
Línea 12 con error
Línea 13 con error
Línea 14 con error
Línea 15 con error
Línea 16 con error
Línea 17 con error
Línea 18 con error
Línea 19 con error
Línea 21 con error
Línea 22 con error
Línea 23 con error
Línea 24 con error
Línea 25 con error
Línea 26 con error
Línea 27 con error
Línea 28 con error
Línea 29 con error
Línea 31 con error
Línea 32 con error
Línea 33 con error
Línea 34 con error
Línea 35 con error
Línea 36 con error
Línea 37 con error
Línea 38 con error
Línea 39 con error
Línea 41 con error
Línea 42 con error
Línea 43 con error
Línea 44 con error
Línea 45 con error
Línea 46 con error
Línea 47 con error
Línea 48 con error
Línea 49 con error
```

## Ejercicio 2

2. Reemplaza todas las ocurrencias de "DEBUG" por "INFO" en el archivo `file1.txt` y guarda el resultado en `output2.txt`.

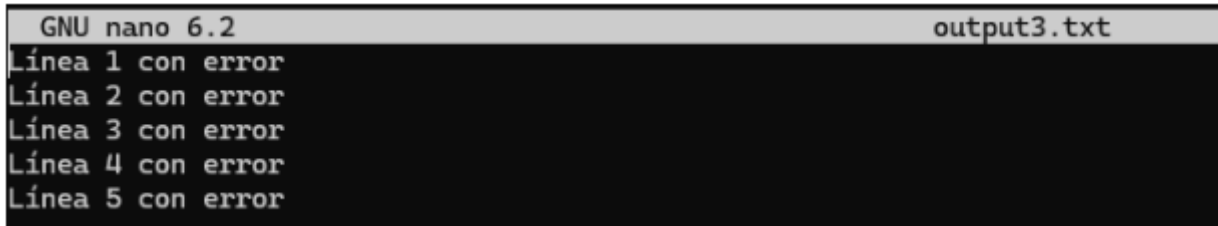
```
sed 's/DEBUG/INFO/g' file1.txt > output2.txt
```



```
GNU nano 6.2                                output2.txt
Línea 1 con error
Línea 2 con error
Línea 3 con error
Línea 4 con error
Línea 5 con error
Línea 6 con error
Línea 7 con error
Línea 8 con error
Línea 9 con error
INFO: Línea de prueba 10
Línea 11 con error
Línea 12 con error
Línea 13 con error
Línea 14 con error
Línea 15 con error
Línea 16 con error
Línea 17 con error
Línea 18 con error
Línea 19 con error
INFO: Línea de prueba 20
Línea 21 con error
Línea 22 con error
Línea 23 con error
Línea 24 con error
Línea 25 con error
Línea 26 con error
Línea 27 con error
Línea 28 con error
Línea 29 con error
INFO: Línea de prueba 30
Línea 31 con error
Línea 32 con error
Línea 33 con error
Línea 34 con error
Línea 35 con error
Línea 36 con error
Línea 37 con error
Línea 38 con error
Línea 39 con error
INFO: Línea de prueba 40
Línea 41 con error
Línea 42 con error
Línea 43 con error
Línea 44 con error
Línea 45 con error
Línea 46 con error
Línea 47 con error
```

3. Muestra las primeras 5 líneas de `file1.txt` que contengan "ERROR" y guarda el resultado en `output3.txt`.

```
head -n 5 file.txt | grep "error" > output3.txt
```



```
GNU nano 6.2 output3.txt
Línea 1 con error
Línea 2 con error
Línea 3 con error
Línea 4 con error
Línea 5 con error
```

#### Ejercicio 4

4. Cuenta el número de archivos `.txt` en el directorio actual y guarda el resultado en `output4.txt`.

```
ls *.txt | wc -l > output4.txt
```



```
GNU nano 6.2 output4.txt
9
```

#### Ejercicio 5

5. Ordena las líneas del archivo `file1.txt` alfabéticamente, elimina duplicados, y guarda el resultado en `output5.txt`.

```
sort file1.txt | uniq > output5.txt
```

```
GNU nano 6.2                                output5.txt
DEBUG: Línea de prueba 10
DEBUG: Línea de prueba 20
DEBUG: Línea de prueba 30
DEBUG: Línea de prueba 40
DEBUG: Línea de prueba 50
Línea 1 con error
Línea 11 con error
Línea 12 con error
Línea 13 con error
Línea 14 con error
Línea 15 con error
Línea 16 con error
Línea 17 con error
Línea 18 con error
Línea 19 con error
Línea 2 con error
Línea 21 con error
Línea 22 con error
Línea 23 con error
Línea 24 con error
Línea 25 con error
Línea 26 con error
Línea 27 con error
Línea 28 con error
Línea 29 con error
Línea 3 con error
Línea 31 con error
Línea 32 con error
Línea 33 con error
Línea 34 con error
Línea 35 con error
Línea 36 con error
Línea 37 con error
Línea 38 con error
Línea 39 con error
Línea 4 con error
Línea 41 con error
Línea 42 con error
Línea 43 con error
Línea 44 con error
Línea 45 con error
Línea 46 con error
Línea 47 con error
Línea 48 con error
Línea 49 con error
Línea 5 con error
Línea 6 con error
```

## Ejercicio 6

6. Divide el archivo `file6_large.txt` en dos archivos: uno con las primeras 100 líneas (`out6.1.txt`) y otro con el resto (`out.6.txt`).

```
head -n 100 file6_large.txt > out6.1.txt | sed '1,100d' file6_large.txt >
out.6.txt
```

```
GNU nano 6.2 out.6.txt
Línea 101
Línea 102
Línea 103
Línea 104
Línea 105
Línea 106
Línea 107
Línea 108
Línea 109
Línea 110
Línea 111
Línea 112
Línea 113
Línea 114
Línea 115
Línea 116
Línea 117
Línea 118
Línea 119
Línea 120
Línea 121
Línea 122
Línea 123
Línea 124
Línea 125
```

```
GNU nano 6.2 out6.1.txt
Línea 1
Línea 2
Línea 3
Línea 4
Línea 5
Línea 6
Línea 7
Línea 8
Línea 9
Línea 10
Línea 11
Línea 12
Línea 13
Línea 14
Línea 15
Línea 16
Línea 17
Línea 18
Línea 19
Línea 20
Línea 21
Línea 22
Línea 23
Línea 24
Línea 25
```

## Ejercicio 7

7. Elimina todas las líneas en blanco del archivo `file4.txt` y guarda el resultado en `output7.txt`.



```
sed '/^$/d' file4.txt | uniq > output7.txt
```



## Ejercicio 8

8. Reemplaza todas las comas por tabulaciones en el archivo `file2.csv` y guarda el resultado en `output8.csv`.

```
sed 's/','/' '\t/g' file2.csv > output8.csv
```

GNU nano 6.2		output8.csv
nombre	edad	ciudad
Persona1	21	Ciudad1
Persona2	22	Ciudad2
Persona3	23	Ciudad3
Persona4	24	Ciudad4
Persona5	25	Ciudad5
Persona6	26	Ciudad6
Persona7	27	Ciudad7
Persona8	28	Ciudad8
Persona9	29	Ciudad9
Persona10	30	Ciudad10
Persona11	31	Ciudad11
Persona12	32	Ciudad12
Persona13	33	Ciudad13
Persona14	34	Ciudad14
Persona15	35	Ciudad15
Persona16	36	Ciudad16
Persona17	37	Ciudad17
Persona18	38	Ciudad18
Persona19	39	Ciudad19
Persona20	40	Ciudad20
Persona21	41	Ciudad21
Persona22	42	Ciudad22
Persona23	43	Ciudad23
Persona24	44	Ciudad24

### Ejercicio 9

9. Inserta el prefijo "INICIO:" antes de cada línea que comience con una vocal en el archivo `file5.txt` y guarda el resultado en `output9.txt`.

```
sed '/^[aeiouAEIOU]/s/^/INICIO:/' file5.txt > output9.txt
```

GNU nano 6.2		output9.txt
INICIO:Una línea 1		
INICIO:Otra línea 2		
INICIO:Ejemplo línea 3		
INICIO:Otra línea 4		

### Ejercicio 10

10. Cuenta cuántas líneas en el archivo `file1.txt` contienen la palabra "ERROR" y guarda el resultado en `output10.txt`.

```
grep "error" file1.txt | wc -l > output10.txt
```

GNU nano 6.2		output10.txt
45		

## Ejercicio 11

11. Crea un script que muestre el número total de archivos en el directorio actual y escribe el resultado en `output11.txt`.

```
ls *.* -A | wc -l
```

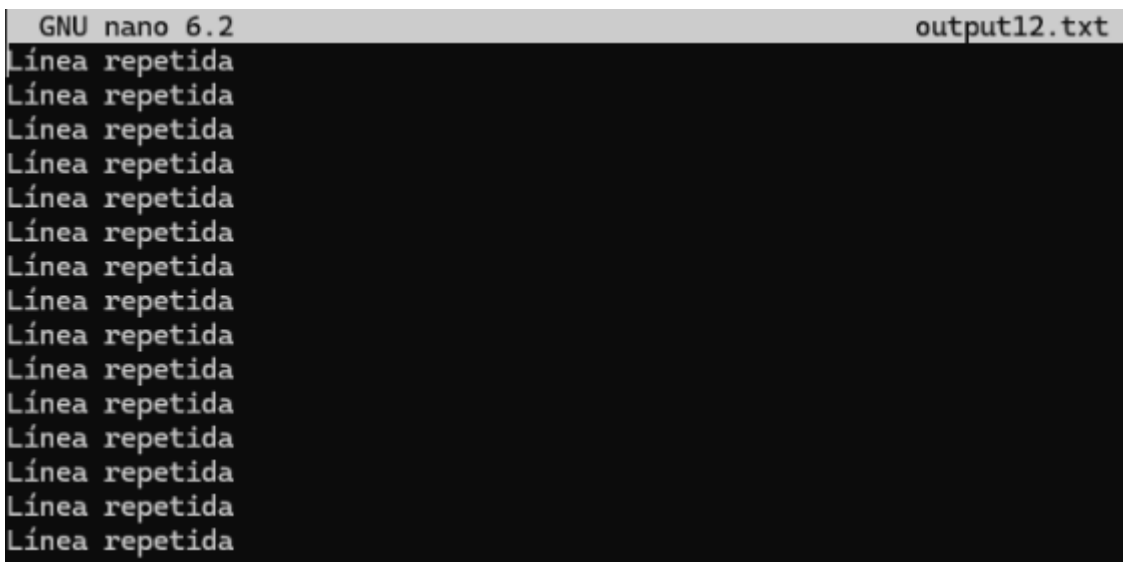


```
GNU nano 6.2 output11.txt
20
```

## Ejercicio 12

12. Encuentra líneas duplicadas en el archivo `file7_duplicates.txt`, muestra cuántas veces se repiten, y guarda el resultado en `output12.txt`.

```
uniq -D file7_duplicates.txt > output12.txt
```



```
GNU nano 6.2 output12.txt
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
Línea repetida
```

## Ejercicio 13

13. Extrae las primeras 100 líneas del archivo `file6_large.txt` y guarda el resultado en `output13.txt`.

```
head -n 100 file6_large.txt > output13.txt
```

```
Línea 56
Línea 57
Línea 58
Línea 59
Línea 60
Línea 61
Línea 62
Línea 63
Línea 64
Línea 65
Línea 66
Línea 67
Línea 68
Línea 69
Línea 70
Línea 71
Línea 72
Línea 73
Línea 74
Línea 75
Línea 76
Línea 77
Línea 78
Línea 79
Línea 80
Línea 81
Línea 82
Línea 83
Línea 84
Línea 85
Línea 86
Línea 87
Línea 88
Línea 89
Línea 90
Línea 91
Línea 92
Línea 93
Línea 94
Línea 95
Línea 96
Línea 97
Línea 98
Línea 99
Línea 100
cfigs@Info1-PC06:~/DAM/Sistemas/Bash/Bash/Ejercicios/Ejercicios a entregar$ |
```

#### Ejercicio 14

14. Busca todas las líneas que contengan números en el archivo `file1.txt` y guarda el resultado en `output14.txt`.

```
grep '[0-9]' file1.txt > output14.txt
```

```
DEBUG: Línea de prueba 20
Línea 21 con error
Línea 22 con error
Línea 23 con error
Línea 24 con error
Línea 25 con error
Línea 26 con error
Línea 27 con error
Línea 28 con error
Línea 29 con error
DEBUG: Línea de prueba 30
Línea 31 con error
Línea 32 con error
Línea 33 con error
Línea 34 con error
Línea 35 con error
Línea 36 con error
Línea 37 con error
Línea 38 con error
Línea 39 con error
DEBUG: Línea de prueba 40
Línea 41 con error
Línea 42 con error
Línea 43 con error
Línea 44 con error
Línea 45 con error
Línea 46 con error
Línea 47 con error
Línea 48 con error
Línea 49 con error
DEBUG: Línea de prueba 50
cfgs@Info1-PC06:~/DAM/Sistemas/Bash/Bash/Ejercicios/Ejercicios a entregar$ grep '[0-9]' file1.txt > output14.txt
```

## Ejercicio 15

15. Convierte el archivo `file2.csv` en un archivo delimitado por punto y coma y guarda el resultado en `output15.csv`.

```
sed 's/$/;/g' file2.csv > output15.csv
```

```
GNU nano 6.2 output15.csv
nombre,edad,ciudad;
Persona1,21,Ciudad1;
Persona2,22,Ciudad2;
Persona3,23,Ciudad3;
Persona4,24,Ciudad4;
Persona5,25,Ciudad5;
Persona6,26,Ciudad6;
Persona7,27,Ciudad7;
Persona8,28,Ciudad8;
Persona9,29,Ciudad9;
Persona10,30,Ciudad10;
Persona11,31,Ciudad11;
Persona12,32,Ciudad12;
Persona13,33,Ciudad13;
Persona14,34,Ciudad14;
Persona15,35,Ciudad15;
Persona16,36,Ciudad16;
Persona17,37,Ciudad17;
Persona18,38,Ciudad18;
Persona19,39,Ciudad19;
Persona20,40,Ciudad20;
Persona21,41,Ciudad21;
Persona22,42,Ciudad22;
Persona23,43,Ciudad23;
Persona24,44,Ciudad24;
Persona25,45,Ciudad25;
Persona26,46,Ciudad26;
Persona27,47,Ciudad27;
Persona28,48,Ciudad28;
Persona29,49,Ciudad29;
Persona30,50,Ciudad30;
```

## Ejercicio 16

16. Inserta un prefijo "MODIFICADO:" antes de las líneas que contengan la palabra "ERROR" en el archivo `file1.txt` y guarda el resultado en `output16.txt`.

```
sed '/error/s/^/MODIFICADO: /' file1.txt > output16.txt
```

```
GNU nano 6.2                                output16.txt
MODIFICADO: Línea 1 con error
MODIFICADO: Línea 2 con error
MODIFICADO: Línea 3 con error
MODIFICADO: Línea 4 con error
MODIFICADO: Línea 5 con error
MODIFICADO: Línea 6 con error
MODIFICADO: Línea 7 con error
MODIFICADO: Línea 8 con error
MODIFICADO: Línea 9 con error
DEBUG: Línea de prueba 10
MODIFICADO: Línea 11 con error
MODIFICADO: Línea 12 con error
MODIFICADO: Línea 13 con error
MODIFICADO: Línea 14 con error
MODIFICADO: Línea 15 con error
MODIFICADO: Línea 16 con error
MODIFICADO: Línea 17 con error
MODIFICADO: Línea 18 con error
MODIFICADO: Línea 19 con error
DEBUG: Línea de prueba 20
MODIFICADO: Línea 21 con error
MODIFICADO: Línea 22 con error
MODIFICADO: Línea 23 con error
MODIFICADO: Línea 24 con error
MODIFICADO: Línea 25 con error
MODIFICADO: Línea 26 con error
MODIFICADO: Línea 27 con error
MODIFICADO: Línea 28 con error
MODIFICADO: Línea 29 con error
DEBUG: Línea de prueba 30
MODIFICADO: Línea 31 con error
MODIFICADO: Línea 32 con error
MODIFICADO: Línea 33 con error
MODIFICADO: Línea 34 con error
MODIFICADO: Línea 35 con error
MODIFICADO: Línea 36 con error
MODIFICADO: Línea 37 con error
MODIFICADO: Línea 38 con error
MODIFICADO: Línea 39 con error
DEBUG: Línea de prueba 40
```

## Ejercicio 17

17. Extrae las líneas del archivo `file3.txt` que contengan números pares y guarda el resultado en `output17.txt`.

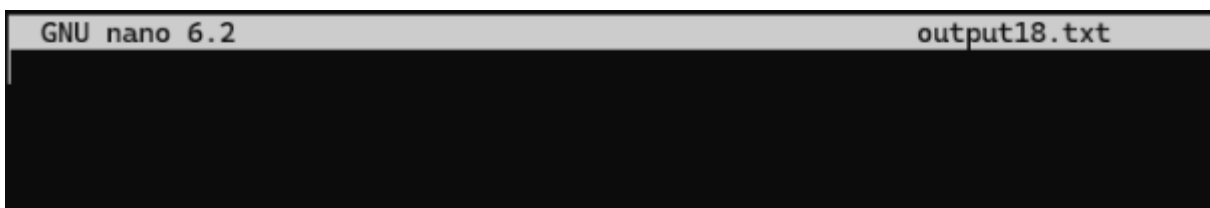
```
grep [02468] file3.txt > output17.txt
```

```
cfgs@Info1-PC06:~/DAM/Sistemas/Bash/Bash/Ejercicios/Ejercicios a entregar$ grep [02468] file3.txt
2
4
6
8
10
12
14
16
18
20
21
22
23
24
25
26
27
28
29
30
```

### Ejercicio 18

18. Busca recursivamente en todos los subdirectorios del directorio actual archivos que contengan la palabra "ERROR" y guarda las coincidencias en `output18.txt`.

```
grep -r "error" ./sub* > output18.txt
```



### Ejercicio 19

19. Extrae las últimas 50 líneas del archivo `file6_large.txt` y guarda el resultado en `output19.txt`.

```
tail -n 50 file6_large.txt > output19.txt
```



```
GNU nano 6.2 output19.txt
Línea 151
Línea 152
Línea 153
Línea 154
Línea 155
Línea 156
Línea 157
Línea 158
Línea 159
Línea 160
Línea 161
Línea 162
Línea 163
Línea 164
Línea 165
Línea 166
Línea 167
Línea 168
Línea 169
Línea 170
Línea 171
Línea 172
Línea 173
Línea 174
Línea 175
Línea 176
Línea 177
Línea 178
Línea 179
Línea 180
Línea 181
Línea 182
Línea 183
Línea 184
Línea 185
Línea 186
Línea 187
Línea 188
Línea 189
Línea 190
Línea 191
Línea 192
Línea 193
Línea 194
Línea 195
Línea 196
```

## Ejercicio 20

20. Lista todos los archivos, incluidos los ocultos, de forma recursiva desde el directorio actual y guarda el resultado en `output20.txt`.

```
find | wc -l > output20.txt
```

```
GNU nano 6.2 output20.txt
41
```