

Hasta ahora hemos visto en las prácticas anteriores cómo ejecutar mandatos en Linux de forma secuencial (ejecutar un mandato, esperar a que el mismo produzca su salida, ejecutar otro a continuación y así sucesivamente). En la práctica de hoy veremos la idea de redireccionamiento de la salida o de la entrada de un mandato. Muchos de los mandatos en Unix tienen un canal o flujo de entrada (stdin), a partir del cual leen la información que debe ser procesada. Algunos mandatos no requieren de una entrada explícita de información (por ejemplo, ls se ejecuta sin entrada adicional). Sin embargo, otros como por ejemplo “grep” (mandato usado para búsqueda de expresiones regulares) sí que requieren de una entrada estándar en donde buscar las expresiones regulares.

Guión de la práctica: Recoge en un documento pantallazos de todo el proceso.

Redireccionando la entrada y salida

1. Inicia la máquina virtual Ubuntu.
2. Crea un directorio llamado “practica21” y haz que sea tu directorio de trabajo.

```
mario@mario-VirtualBox:~$ mkdir practica21
mario@mario-VirtualBox:~$ cd practica21/
```

3. Lee el siguiente documento: http://tallernulinux.sourceforge.net/practicos/TP4_stdio.pdf

4. Ejecuta el mandato “cat”. ¿Qué ha sucedido con el prompt? ¿Qué está esperando el intérprete de mandatos? Escribe cualquier frase y después pulsa “Enter”. ¿Qué ha sucedido? Escribe nuevas frases (recuerda pulsar “Enter” para terminar cada frase. ¿Qué sucede?

Al pulsar cat, simplemente esta esperando a que le introduzcas algo, por eso se queda así:

```
mario@mario-VirtualBox:~/practica21$ cat
```

Hasta que no le des a control+c no saldrá de ahí por mucho intro que aprietes

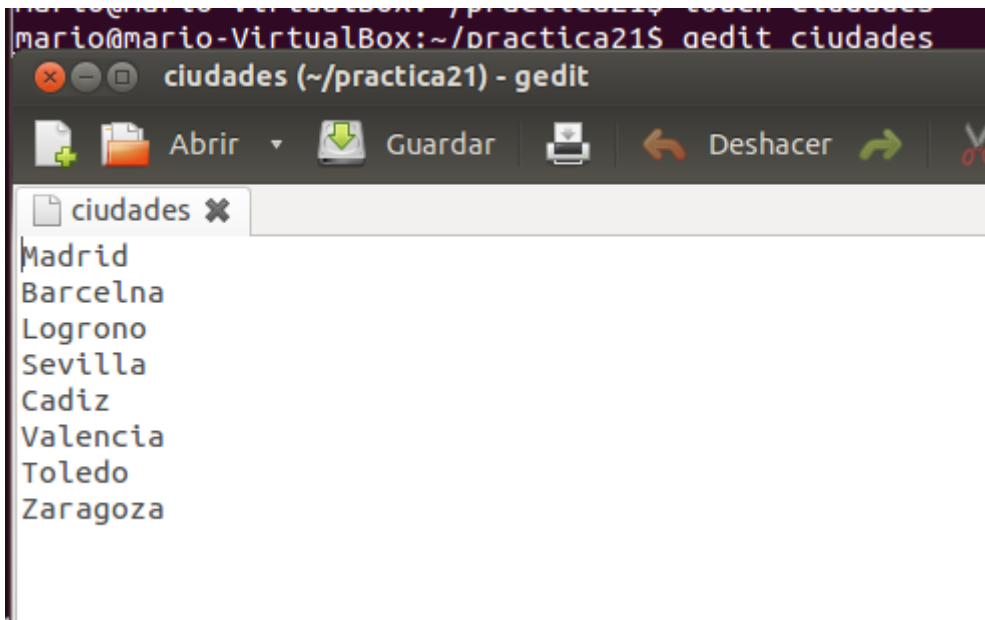
5. El mandato “cat” se utiliza para concatenar uno o más ficheros (concatenar un único fichero es equivalente a crear un fichero igual a ese). Su sintaxis es “cat fichero1 fichero2 fichero3”. Por defecto, el resultado de concatenar los ficheros se muestra en la salida estándar de la shell. Como no le hemos dicho cuál era su entrada estándar (stdin) de información (por ejemplo, un fichero) ha tomado por defecto como entrada estándar el teclado de la terminal. Detén el mandato “cat” pulsando Ctrl+D. Esta combinación de teclas le envía al proceso una señal de fin de fichero (EOF). Al igual que un flujo de entrada por defecto (stdin), los mandatos también tienen un flujo de salida (stdout). Por ejemplo, con el mandato “cat” anterior, cada vez que se leía una línea, se mostraba de nuevo esa línea en la misma shell. Por tanto, como has podido comprobar, el flujo de salida por defecto de este mandato (y en general de los mandatos en Linux) es la pantalla del terminal. Los dos operadores que nos permiten redirigir la salida de un mandato (y por salida entendemos tanto la salida estándar como la salida de errores que veremos después) son “>” y “>>”. La diferencia entre “>” y “>>” es que el primero (“>”) siempre crea un nuevo fichero conteniendo la información volcada (y si existía un fichero con ese nombre lo “pisa”), mientras que el segundo (“>>”) crea un fichero con la salida, o si el fichero ya existía simplemente añade su salida a la información que ya hubiese en el fichero.

6. Veamos ahora una nueva forma de generar un fichero de texto siguiendo estas ideas. Ejecuta el mandato “cat > ciudades” (es equivalente a “cat 1> ciudades”). El intérprete espera que le demos la entrada para el mandato “cat”. La salida del mandato se hará al fichero “ciudades”. Como no le hemos dado ningún fichero al mandato cat para concatenar, por defecto considera la propia shell como “fichero” de entrada.

```
mario@mario-VirtualBox:~/practica21$ cat > ciudades
^?^C
mario@mario-VirtualBox:~/practica21$ ls
ciudades
```

7. Escribe el nombre de 8 ciudades, pulsando “Enter” después de cada una de ellas. (Pulsa “Ctrl + D” para terminar la operación). Comprueba, por ejemplo con “less ciudades”, que el fichero se ha creado correctamente y que contiene la información que esperabas.

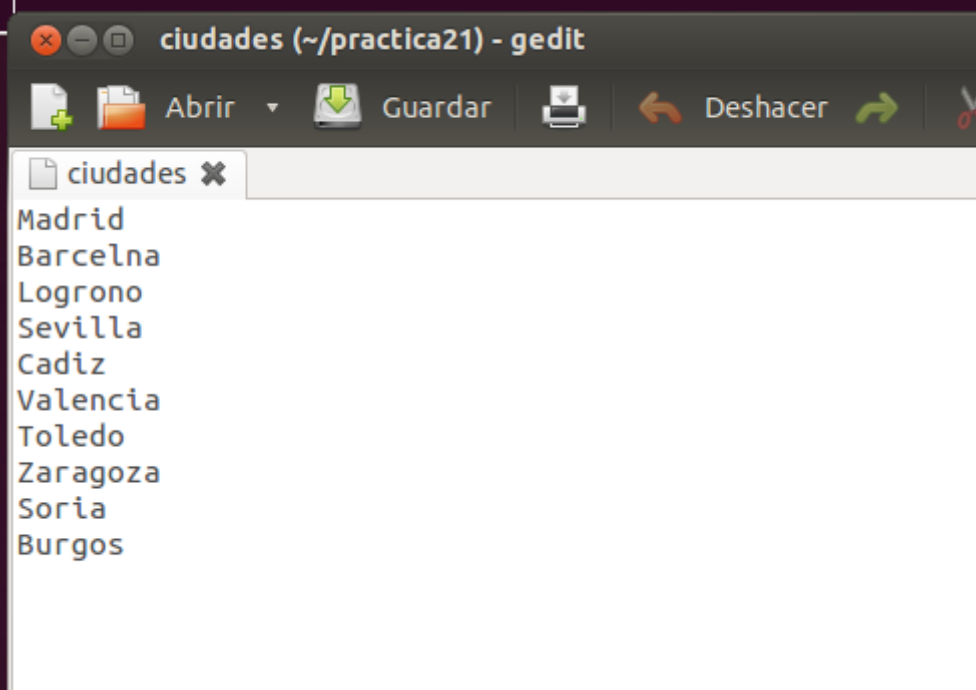
```
mario@mario-VirtualBox:~/practica21$ cat > ciudades
Madrid
Barcelona
Logrono
Sevilla
Cadiz
Valencia
Toledo
Zaragoza
^D
mario@mario-VirtualBox:~/practica21$ less ciudades
```



8. Ejecuta el mandato “cat >> ciudades”. Introduce el nombre de otras dos ciudades (pulsando Enter” entre medio y “Ctrl + D” para terminar). ¿Cuál es el contenido del fichero ahora? ¿Se ha borrado la información anterior del fichero?

```
mario@mario-VirtualBox:~/practica21$ cat >> ciudades
Soria
Burgos^C
```

```
mario@mario-VirtualBox:~/practica21$ gedit ciudades
```

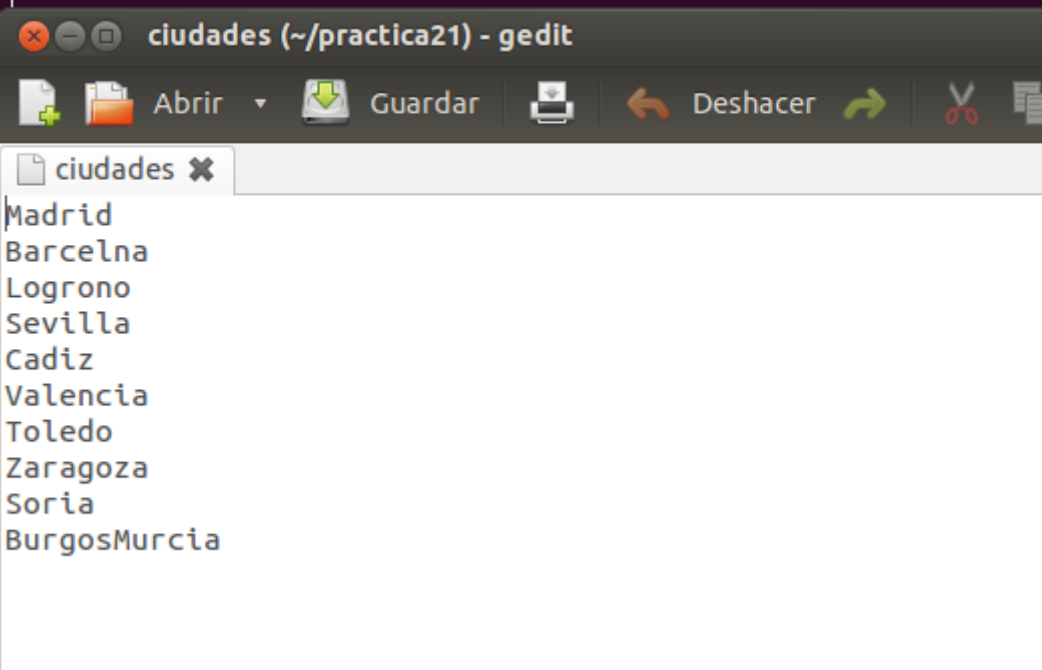


The screenshot shows a terminal window with the command `gedit ciudades` executed. Below the terminal, a window titled "ciudades (~/.practica21) - gedit" is open. The window has a menu bar with "Abrir", "Guardar", "Deshacer", and a scissors icon. The main text area contains a list of cities: Madrid, Barcelona, Logrono, Sevilla, Cadiz, Valencia, Toledo, Zaragoza, Soria, and Burgos.

9. Otro mandato que nos permite mandar mensajes es "echo". Ejecuta el siguiente mandato: (echo "El usuario activo es \$USER") ¿Cuál es el resultado obtenido? De nuevo, la salida estándar para "echo" es la propia shell, pero esto se puede cambiar por medio de la redirección de su salida: (echo "Murcia" >> ciudades) Comprueba el contenido del fichero "ciudades" de nuevo. ¿Cuál es ahora el contenido?

```
mario@mario-VirtualBox:~/practica21$ echo "el usuario activo es $USER"
el usuario activo es mario
```

```
mario@mario-VirtualBox:~/practica21$ echo Murcia >>ciudades
mario@mario-VirtualBox:~/practica21$ gedit ciudades
```



The screenshot shows a terminal window with the command `echo "el usuario activo es $USER"` executed, resulting in the output "el usuario activo es mario". Below the terminal, a window titled "ciudades (~/.practica21) - gedit" is open. The window has a menu bar with "Abrir", "Guardar", "Deshacer", and a scissors icon. The main text area contains a list of cities: Madrid, Barcelona, Logrono, Sevilla, Cadiz, Valencia, Toledo, Zaragoza, Soria, Burgos, and Murcia.

10. Como hemos ido viendo en las prácticas anteriores, en Linux la mayor parte de la información del sistema se gestiona por medio de ficheros de texto; es por este motivo que existen múltiples mandatos que nos permiten tratar este tipo de ficheros:

11. Ejecuta el siguiente mandato. ¿Cuál ha sido el resultado obtenido? (sort ciudades)

Te las ordena alfabéticamente de a-z

```
mario@mario-VirtualBox:~/practica21$ sort ciudades
Barcelna
BurgosMurcia
Cadiz
Logrono
Madrid
Sevilla
Soria
Toledo
Valencia
Zaragoza
```

12. Ejecuta el siguiente mandato. ¿Cuál ha sido el resultado obtenido? (sort -r ciudades)

Te las ordena alfabéticamente de z-a (a la inversa)

```
mario@mario-VirtualBox:~/practica21$ sort -r ciudades
Zaragoza
Valencia
Toledo
Soria
Sevilla
Madrid
Logrono
Cadiz
BurgosMurcia
Barcelna
```

13. Ejecuta el mandato: (sort ciudades > ciudades_ordenadas). Comprueba que el fichero “ciudades_ordenadas” contiene la misma información que el fichero ciudades pero ordenadas alfabéticamente.

```
mario@mario-VirtualBox:~/practica21$ sort ciudades >ciudades_ordenadas
mario@mario-VirtualBox:~/practica21$ cat ciudades_ordenadas
Barcelona
BurgosMurcia
Cadiz
Logrono
Madrid
Sevilla
Soria
Toledo
Valencia
Zaragoza
mario@mario-VirtualBox:~/practica21$
```

14. Crea, con el mandato “cat”, un fichero llamado frutas con el nombre de 5 frutas. Crea, con el mandato “cat”, otro fichero llamado colores con el nombre de 4 colores. Concatena los ficheros frutas y colores y redirige la salida a un fichero con el nombre “frutas_y_colores”.

```
mario@mario-VirtualBox:~/practica21$ cat > frutas
Sandia
Melocotn
Melon
Platano
Manzana
^C
```

```
mario@mario-VirtualBox:~/practica21$ cat > colores
naranja
rojo
azul
amarillo
violeta
^C
```

```
Manzana
mario@mario-VirtualBox:~/practica21$ cat colores>>frutas_colores
mario@mario-VirtualBox:~/practica21$ cat frutas >> frutas_colores
```

```
mario@mario-VirtualBox:~/practica21$ cat frutas_colores
Sandia
Melocotn
Melon
Platano
Manzana
naranja
rojo
azul
amarillo
violeta
```

15. Comprueba la utilidad del mandato wc (por ejemplo, por medio de man). Comprueba el número de caracteres del fichero “frutas_y_colores”. Comprueba su número de palabras. Comprueba su número de líneas. (wc frutas_y_colores)

```
mario@mario-VirtualBox:~/practica21$ wc frutas_colores
10 10 73 frutas_colores
```

16. Aparte de un flujo estándar de entrada (por defecto la shell) y de un flujo estándar de salida (de nuevo, por defecto la shell), cualquier mandato puede producir también una salida de errores (conocida como stderr). En general estos errores nos mostrarán información sobre operaciones que no han podido ser completadas con éxito (por ejemplo, por no disponer de permisos, por falta de recursos del ordenador, ...). Ejecuta el siguiente mandato: (cat inexistente) ¿Cuál es el resultado? La salida que has obtenido es lo que se conoce como salida de error. Por defecto, también es volcada a la propia shell donde nos encontramos ejecutando el programa, pero existen formas de redirigirla que veremos a lo largo de la práctica.

```
mario@mario-VirtualBox:~/practica21$ cat inexistente
cat: inexistente: No existe el archivo o el directorio
```

17. Cada uno de los tres flujos anteriores de información (entrada estándar o stdin, salida estándar o stdout y salida de error o stderr) reciben un valor numérico que nos permite referirnos a ellos: stdin es el flujo 0, stdout es el flujo 1 y stderr es el flujo 2. Es importante que retengas la idea de que cada mandato en Linux puede contar con un flujo de entrada, uno de salida y uno de error.

18. Vuelve a repetir la operación que hicimos con anterioridad sobre el fichero “inexistente”, pero redirigiendo ahora la salida de errores a un fichero de nombre “resultado”. (cat inexistente 2> resultado) La sintaxis del mandato anterior debería entenderse como:

“realiza la operación cat resultado, y su segundo flujo de información generado, es decir stderr, redirígelo al fichero resultado”. Comprueba el contenido del fichero “resultado”.

```
mario@mario-VirtualBox:~/practica21$ cat inexistente 2> resultado
mario@mario-VirtualBox:~/practica21$
```

```
mario@mario-VirtualBox:~/practica21$ cat resultado
cat: inexistente: No existe el archivo o el directorio
```

Tuberías

Aparte de las utilidades anteriores, existe otro operador de control en la shell de Linux que nos permite redirigir la salida de un mandato (del modo como hemos hecho con ">" y ">>") para que se convierta en la entrada de un nuevo mandato. Hasta ahora hemos visto varios ejemplos de cómo hacer lo mismo pasando la información por un fichero intermedio. Por medio de lo que se conoce como interconexiones o tuberías (del inglés pipe) podemos conectar dos mandatos, haciendo que la salida de un mandato se convierta directamente en la entrada de otro. El carácter que se utiliza para crear una tubería que redirija la salida de un mandato a la entrada de otro es "|". Además, se pueden interconectar varios mandatos con sucesivas tuberías.

19. Prueba la salida del siguiente mandato: `(cat frutas | wc -l)` ¿Qué resultado has obtenido? ¿Cuál ha sido en este caso el flujo de entrada para el mandato "wc"?

```
mario@mario-VirtualBox:~/practica21$ cat frutas | wc -l
5
```

Mandatos grep

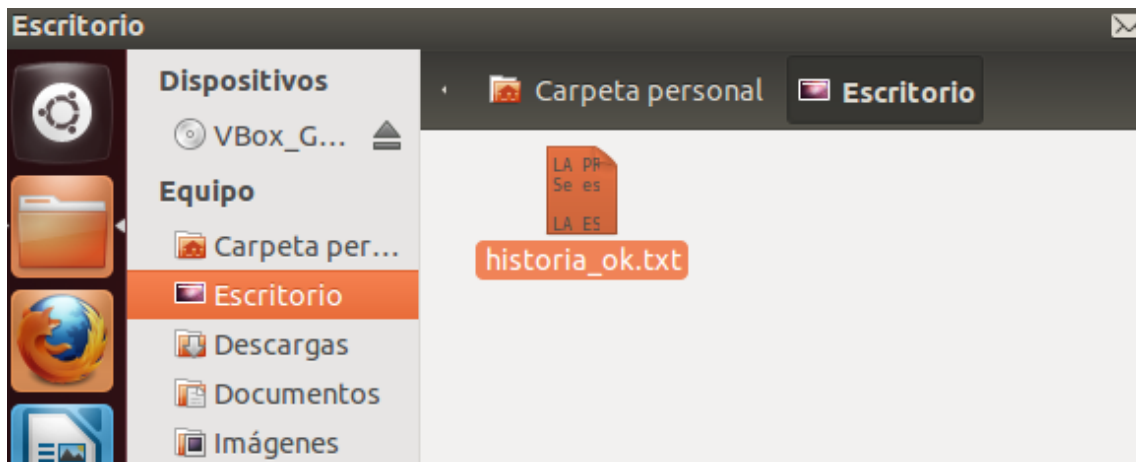
Otro mandato útil para trabajar con textos es el mandato "grep". El mandato "grep" busca dentro de un texto todas las líneas que coincidan con la expresión o patrón que nosotros le indiquemos. Para poder hacer esas búsquedas o modificaciones de texto, se utilizan las expresiones regulares. Una

expresión regular es una forma de describir una (o múltiples) cadenas de texto. La cadena de texto que estamos buscando se debe corresponder con esa expresión regular. Por tanto, un buen manejo de las expresiones regulares nos va a permitir hacer búsquedas de forma más rápida o precisa. En la página <http://www.panix.com/~elflord/unix/grep.html> (en inglés) puedes encontrar un buen tutorial para aprender expresiones regulares. La página del manual de grep (man grep) en su sección "Regular Expressions" también contiene información de utilidad. Una versión

extendida de grep es el mandato egrep (extended grep). Se caracteriza porque su lenguaje de expresiones regulares es más expresivo que el propio de "grep".

20. Realiza algunos de los ejercicios (nivel medio) de <http://lem.eui.upm.es/tgrep.html>. Indica con tus palabras qué hace cada expresión aunque no des todos los resultados posibles del ejercicio en concreto. En <http://lem.eui.upm.es/tgrepApun.html> de esa página tienes la explicación de los caracteres.

21. Copia a tu escritorio el fichero historia.txt



22. Muestra todas las líneas que contengan la cadena "España". (cat historia | grep "España")

23. Por medio de "wc" comprueba que la palabra "España" aparece en ¿cuántas líneas?. (cat historia | grep "España" | wc -l)

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "España" | wc -l
8
```

24. Muestra todas las líneas que contengan primero la cadena "España" y luego la cadena "guerra". (cat historia | grep "España.*guerra"). Por medio de "wc" comprueba que la expresión regular aparece solo en ? línea.

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "España.*guerra"
mario@mario-VirtualBox:~/Escritorio$
```

25. Muestra las líneas que contengan primero la cadena "guerra" y luego la cadena "España". Por medio de "wc" comprueba que aparece en ? líneas.

```
Juan Carlos de Borbón, nieto de Alfonso XIII, príncipe de España, su sucesor a título de
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "España" | grep "guerra"
```


26. Muestra las líneas que contengan la cadena “España” y “guerra” en cualquier orden. (cat historia | grep “España” | grep “guerra”)

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "España" | grep "guerra"
España siguió prosperando bajo la dinastía Hasburgo gracias al comercio con las colonias americanas
, pero al mismo tiempo sostuvo guerras contra Francia, Holanda e Inglaterra, culminando con la desas-
trrosa derrota de la "Armada Invencible" en 1.588. Cuando el último rey de la dinastía de los Hasbu-
rgo murió sin descendencia, Felipe de Borbón, sobrino del rey de Francia, Luis XIV, le sucedió en e-
l trono. Como consecuencia de la Revolución Francesa, España declaró la guerra a la nueva república
, pero fue derrotada. Napoleón tomó el poder y envió sus tropas contra España en 1808, imponiendo a
su hermano José en el trono. Los españoles mantuvieron una Guerra de Independencia que duraría 5 a-
ños. Tras la derrota definitiva de Napoleón en Waterloo, en 1815, Fernando VII vuelve al trono de E-
spaña y comienza un sistema de rígido absolutismo. Como consecuencia de la designación como hereder-
a de su hija Isabel II, mediante la derogación de la Ley Sálica que impedía la sucesión real de muj-
eres, su hermano Carlos se rebela contra ello iniciándose la Guerra de los Siete Años. La recesión
económica y la inestabilidad política fueron lógicas consecuencias tras la guerra, y España perdió
sus colonias de ultramar, con la excepción de Puerto Rico, Cuba y Filipinas. La revolución de 1868
obligó a Isabel II a renunciar al trono. Se convocaron Cortes Constituyentes que se pronunciaron po-
r el régimen monárquico y se ofreció la corona a Amadeo de Saboya, hijo del rey de Italia. Su breve
reinado dio paso a la proclamación de la I República, que tampoco gozó de larga vida, con el Golpe
de Estado del General Pavia que disolvió el Parlamento. Con ello se proclama rey a Alfonso XII, hi-
jo de Isabel II. En 1885 murió Alfonso XII y se encargó la regencia a su viuda Maria Cristina, hast-
a la mayoría de edad de su hijo Alfonso XIII. La rebelión en 1895 de Cuba en pro de la independenci-
a, decide a los Estados Unidos a declarar la guerra a España, con su derrota España perdió sus últi-
mas colonias en ultramar.
El siglo comienza con una gran crisis económica y la subsiguiente inestabilidad política que desemb-
ocó en el Golpe de Estado del general Primo de Rivera, que estableció una dictadura militar hasta 1
930, año en el que presentó su dimisión al rey y se marchó a París, lugar donde murió. Las elecciones
```

27. Por medio de “wc” comprueba que aparece en ? líneas. Muestra las líneas que contengan la cadena “guerra” pero no “España” (la opción grep -v hace búsquedas inversas, es decir, todas las líneas que no contengan la expresión regular especificada). Comprueba que la expresión buscada aparece en ? líneas.

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep -v "España" | grep "guerra"
Juan Carlos de Borbón, nieto de Alfonso XIII, príncipe de España, su sucesor a título de
```

28. Muestra las líneas que contengan números de cuatro o más cifras (puedes crear intervalos o rangos de valores por medio de [0-9], o también puedes usar la clase de caracteres [:digit:], que representa cualquier dígito, es decir, equivale a [0-9]). (cat historia | grep “[0-9][0-9][0-9][0-9]”). Por medio de “wc” comprueba que aparecen en ? líneas.

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "[0-9][0-9][0-9][0-9]"
```

29. Muestra las líneas que empiecen con “a” (el carácter de principio de línea en expresiones regulares es “^”) (cat historia | grep “^a”) Por medio de “wc” comprueba que son ? líneas.

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "^a"
```

30. Muestra las líneas que terminen con “s” (el carácter para fin de línea en expresiones regulares es “\$”). (cat historia | grep “s\$”). Por medio de “wc” comprueba que son ? líneas

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "s$"
```

31. Muestra las líneas que empiecen con “a” y terminen con “s”. (cat historia | grep “^a.*s\$”) Por medio de “wc” comprueba que solo hay una línea.

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "^a.*s$"
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "^a.*s$"
```

32. Muestra las líneas que contengan fechas relativas al siglo XX.(cat historia | grep "19[0-9][0-9]") Por medio de una tubería y "wc" comprueba que son ? líneas

```
mario@mario-VirtualBox:~/Escritorio$ cat historia_ok.txt | grep "19[0-9][0-9]"
```