

Universidad Tecnológica Nacional - Facultad Regional Buenos Aires Sintaxis y Semántica de los Lenguajes

2021

Trabajo Práctico Individual

Curso: K2054

Profesor: Pablo Damián Mendez

<u>Tema:</u> Expresiones regulares en Bash

Fecha de entrega: 3/06/2021

Legajo: 167.926-0

<u>Correo institucional</u>: mdaguerre@fbra.utn.edu.ar

Usuario github: mdaguerre95

Link repo: https://github.com/mdaguerre95/sintaxis-tp1.git

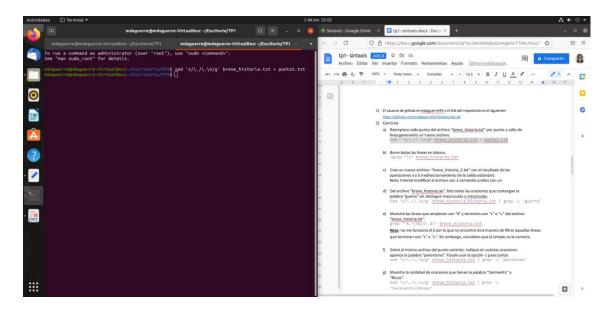
Nombre y apellido: Micaela Daguerre

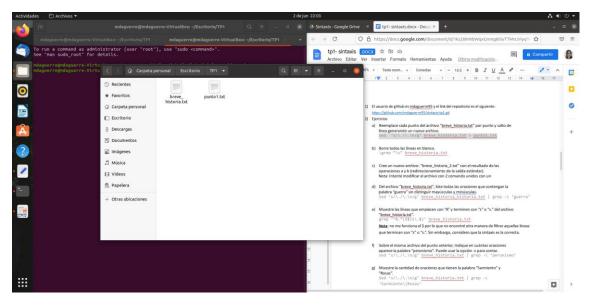
Consigna

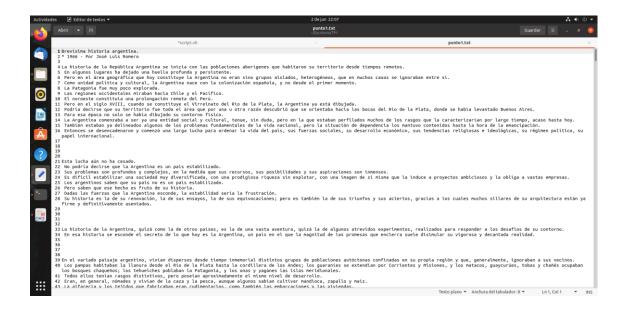
- 1) Crear un usuario en GitHub https://github.com/ con el correo institucional frba. Crear un repositorio. Dentro del repositorio deberá subir todos los archivos que compongan la entrega de este trabajo dentro de una carpeta llamada "TP 1". Para desarrollar este punto, ver el apéndice de este documento.
- 2) Debe entregar un único script que resuelva los siguientes puntos:
 - a) Reemplace cada punto del archivo "breve_historia.txt" por punto y salto de línea generando un nuevo archivo.
 - b) Borre todas las líneas en blanco.
 - c) Cree un nuevo archivo: "breve_historia_2.txt" con el resultado de las operaciones a y b (redireccionamiento de la salida estándar).
 - d) Del archivo "breve_historia.txt", liste todas las oraciones que contengan la palabra "guerra" sin distinguir mayúsculas y minúsculas.
 - e) Muestre las líneas que empiecen con "A" y terminen con "s" o "s." del archivo "breve historia.txt".
 - f) Sobre el mismo archivo del punto anterior, Indique en cuántas oraciones aparece la palabra "peronismo". Puede usar la opción -c para contar.
 - g) Muestre la cantidad de oraciones que tienen la palabra "Sarmiento" y "Rosas".
 - h) Muestre las oraciones que tengan fechas referidas al siglo XIX.
- i) Borre la primera palabra de cada línea. Utilice substitución con sed. La sintaxis para sustituir la primera palabra de cada línea por "nada" sería: \$sed "s/^[[a-zA-Z]]*\b//g" nombre_archivo
- (La "s" indica sustitución; entre los dos primeros /.../ está la expresión regular que queremos reemplazar, en este caso "/^[[a-zA-Z]]*\b"; entre el segundo y el tercer "//" se indica la expresión por la cual será reemplazada, en este caso por la palabra vacía. Finalmente la "g" indica que el cambio será en todo el archivo.
- j. Enumere todos los archivos de una carpeta que contengan extensión ".txt". (tip: pipe con el comando ls).
- 3. Investigue y explique, dando ejemplos cómo se utilizan los siguientes elementos en bash:
- Variables.
- Sentencias condicionales.
- Sentencias cíclicas.
- Subprogramas

Dé ejemplos de cada una.

- 1) El usuario de github es mdaguerre95 y el link del repositorio es el siguiente: https://github.com/mdaguerre95/sintaxis-tp1.git
- 2) Ejercicios
 - a) Reemplace cada punto del archivo "breve_historia.txt" por punto y salto de línea generando un nuevo archivo.
 - sed 's/\./\n/g' breve_historia.txt > punto1.txt

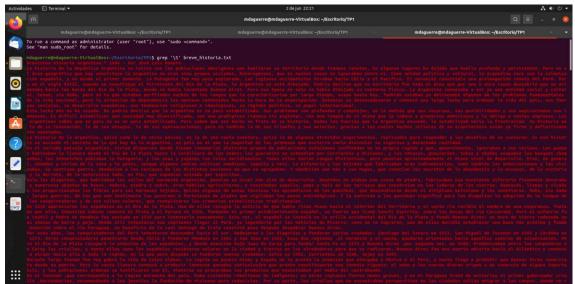






b) Borre todas las líneas en blanco.

\grep "\S" breve_historia.txt

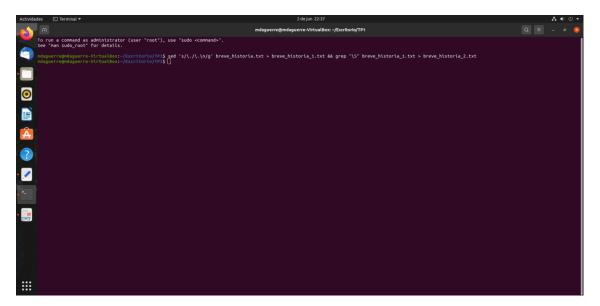


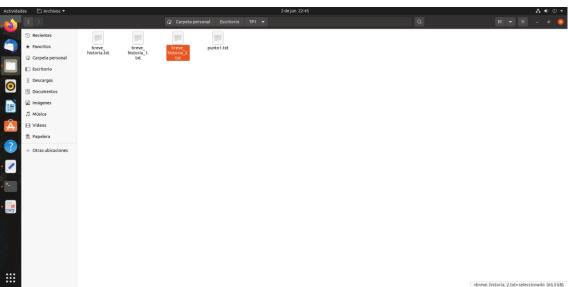
c) Cree un nuevo archivo: "breve_historia_2.txt" con el resultado de las operaciones a y b (redireccionamiento de la salida estándar).

Nota: Intenté modificar el archivo con 2 comandos unidos con un && pero no me f funcionó. Había optado por esta opción debido a que una vez cumplido el primer comando y generado el archivo, el segundo comando solo debía modificarlo. sed 's\\.\n/g' breve_historia.txt > breve_historia_2.txt && grep "\S" > breve_historia_2.txt

Sin embargo, el archivo quedo vacío por lo que no me funcionó. Debido a esto decidí utilizar un primer archivo auxiliar y modificarlo con el segundo comando generando un nuevo archivo con ambos comandos aplicados.

sed 's/\./\.\n/g' breve_historia.txt > breve_historia_1.txt && grep "\S" breve_historia_1.txt > breve_historia_2.txt

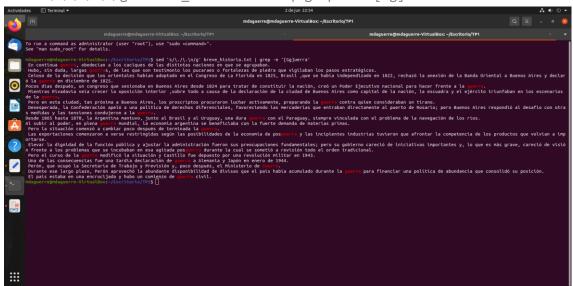






d) Del archivo "breve_historia.txt", liste todas las oraciones que contengan la palabra "guerra" sin distinguir mayúsculas y minúsculas.

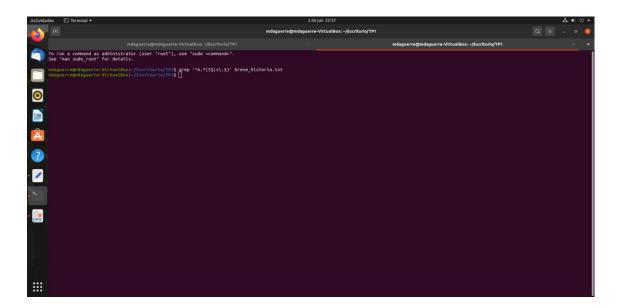
sed 's/\./\.\n/g' breve_historia.txt | grep -e '[Gg]uerra'



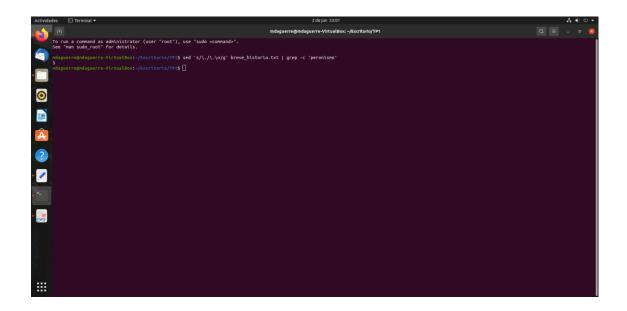
e) Muestre las líneas que empiecen con "A" y terminen con "s" o "s." del archivo "breve_historia.txt".

```
grep '^A.*(s$|s\.$)' breve_historia.txt
```

<u>Nota</u>: no me funciona el \$ por lo que no encontré otra manera de filtrar aquellas líneas que terminan con "s" o "s.". Sin embargo, considero que la sintaxis es la correcta.

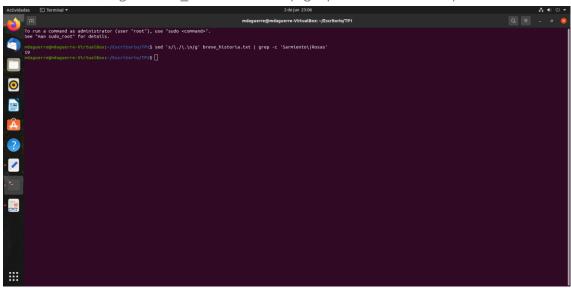


f) Sobre el mismo archivo del punto anterior, Indique en cuántas oraciones aparece la palabra "peronismo". Puede usar la opción -c para contar. sed 's/\.\n/g' breve_historia.txt | grep -c 'peronismo'



g) Muestre la cantidad de oraciones que tienen la palabra "Sarmiento" y "Rosas".

Sed 's/\./\.\n/g' breve_historia.txt | grep -c 'Sarmiento\|Rosas'



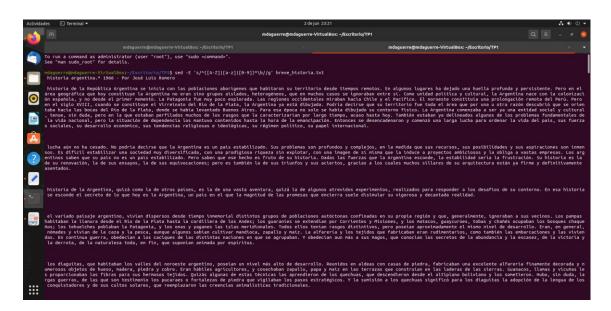
h) Muestre las oraciones que tengan fechas referidas al siglo XIX. Grep -E '180[1-9]|18[1-9][0-9]|1900' breve_historia.txt



i) Borre la primera palabra de cada línea. Utilice sustitución con sed.

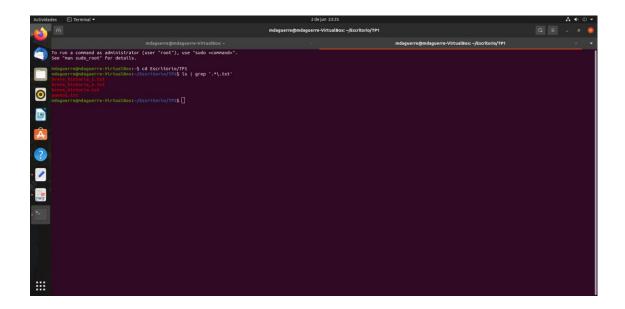
sed -E 's/ $([A-Z]|[a-z]|[0-9])*\b//g'$ breve_historia.txt

<u>Nota</u>: la solución solo la línea del primer párrafo como línea. No encontré la manera de que considere todas las líneas del texto para poder borrar la primera palabra de cada línea.



j) Enumere todos los archivos de una carpeta que contengan extensión ".txt". (tip: pipe con el comando ls).

ls | \grep '.*\.txt'



- 3) Investigue y explique, dando ejemplos cómo se utilizan los siguientes elementos en Bash.
- Variables.
- Sentencias condicionales.
- Sentencias cíclicas.
- Subprogramas

Dé ejemplos de cada una.

Variables

Las variables de bash son como las variables de cualquier lenguaje de programación; es decir, las utilizamos para guardar datos en memoria del ordenador hasta el momento en el cual deseemos utilizarlas. Para asignar un valor a una variable en bash se utiliza el signo "=" como en C y C++. Para recuperar el valor de una variable se utiliza el signo "\$" antes del nombre de la variable.

Las variables del intérprete BASH pueden considerarse desde los siguientes puntos de vista:

- Las variables locales son definidas por el usuario y se utilizan únicamente dentro de un bloque de código, de una función determinada o de un guión.
- Las variables de entorno son las que afectan al comportamiento del intérprete y al de la interfaz del usuario.
- Los parámetros de posición son los recibidos en la ejecución de cualquier programa o función, y hacen referencia a su orden ocupado en la línea de mandatos.
- Las variables especiales son aquellas que tienen una sintaxis especial y que hacen referencia a valores internos del proceso. Los parámetros de posición pueden incluirse en esta categoría

Ejemplo:

to_print='Hola mundo'
echo \$to_print
to_print=5.5
echo \$to print

En el ejemplo anterior le asignamos un valor a la variable y luego le modificamos dicho valor y mostramos ambos valores por pantalla.

• Sentencias condicionales

La sentencia de un condicional es la siguiente:

```
if [[ CONDICIÓN ]];
then
COMANDO 1 si se cumple la condición
fi
```

Además se puede agregar que se debe hacer si la condición no se cumple con la siguiente sintaxis:

Y por ultimo también se permite la utilización de if anidados en bash.

Ejemplo:

```
string1='reo'
if [[ $string1 = *e* ]];
then
echo Eso es verdad
else
echo Eso es mentira
fi
```

En el ejemplo anterior se utiliza los wildcards para realizar comparaciones de cadenas

Condicionales que pueden utilizarse con números:

operador	significado
-lt	menor que (<)
-gt	mayor que (>)
-le	menor o igual que (<=)
-ge	mayor o igual que (>=)
-eq	igual (==)
-ne	no igual (!=)

Condicionales que pueden utilizarse con cadenas:

operador	significado
=	igual, las dos cadenas de texto son exactamente idénticas
!=	no igual, las cadenas de texto no son exactamente idénticas
<	es menor que (en orden alfabético ASCII)
>	es mayor que (en orden alfabético ASCII)
-n	la cadena no está vacía
-Z	la cadena está vacía

Sentencias cíclicas

La sintaxis correspondiente a las sentencias cíclicas o bucles es la siguiente:

```
for VARIABLE in LISTA_VALORES;
do
```

Subprogramas

En Bash para declarar una función se debe escribir la palabra reservada *function*. Puede utilizarse como no la cláusula return para terminar la ejecución. Además, puede haber variables declaradas con la cláusula *local*; estas variables tienen un ámbito de operación interno a la función. El resto de las variables pueden utilizarse en cualquier punto de todo el programa.

La sintaxis es la siguiente de las declaraciones de funciones es la siguiente:

Mientras que la sintaxis de las llamadas a las funciones es la siguiente:

```
NombreFunción [Parámetro1 ...]
```

Ejemplo:

Declaración:

```
welcome()
{
     Echo "welcome"
}
```

Llamado

welcome