UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería en Sistemas de Información

Sintaxis y Semántica de los Lenguajes

Curso K2054

TRABAJO PRÁCTICO N° 1

RegEx en Bash

Alumno: Cesar Mejia
Legajo: 1418713
Correo: cmejia@frba.utn.edu.ar
Usuario Github: comejia
Repositorio: https://github.com/comejia/utn-ssl-examples

Docente: Ing. Pablo D. Mendez

Fecha de entrega:		Nota:		Fecha Aprobación:	

Comentarios:

Tabla de contenido

Consignas

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 “TP1”.
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.

Desarrollo punto 2

Nota: Para todos los puntos se va a mostrar solo parte de la salida de la consola

a) sed 's/\./\.\n/g' breve_historia.txt

```
20:47 $ sed 's/\./\.\n/g' breve_historia.txt
Brevísima historia argentina.
* 1966 - Por José Luis Romero

La historia de la República Argentina se inicia con las poblaciones aborígenes q
ron su territorio desde tiempos remotos.
 En algunos lugares ha dejado una huella profunda y persistente.
 Pero en el área geográfica que hoy constituye la Argentina no eran sino grupos
 heterogéneos, que en muchos casos se ignoraban entre sí.
 Como unidad política y cultural, la Argentina nace con la colonización española
sde el primer momento.
 La Patagonia fue muy poco explorada.
 Las regiones occidentales miraban hacia Chile y el Pacífico.
 El noroeste constituía una prolongación remota del Perú.
 Pero en el siglo XVIII, cuando se constituye el Virreinato del Río de la Plata,
tina ya está dibujada.
 Podría decirse que su territorio fue toda el área que por una u otra razón desc
```

b) sed 's/\./\.\n/g' breve_historia.txt | sed '/^\s*$/d'

```
20:51 $ sed 's/\./\.\n/g' breve_historia.txt | sed '/^\s*$/d'
Brevísima historia argentina.
* 1966 - Por José Luis Romero
La historia de la República Argentina se inicia con las poblaciones aborígenes
ron su territorio desde tiempos remotos.
 En algunos lugares ha dejado una huella profunda y persistente.
 Pero en el área geográfica que hoy constituye la Argentina no eran sino grupos
 heterogéneos, que en muchos casos se ignoraban entre sí.
 Como unidad política y cultural, la Argentina nace con la colonización española
sde el primer momento.
 La Patagonia fue muy poco explorada.
 Las regiones occidentales miraban hacia Chile y el Pacífico.
 El noroeste constituía una prolongación remota del Perú.
 Pero en el siglo XVIII, cuando se constituye el Virreinato del Río de la Plata
tina ya está dibujada.
 Podría decirse que su territorio fue toda el área que por una u otra razón desc
```

c) sed 's/\./\.\n/g' breve_historia.txt | sed '/^\s*$/d' > breve_historia_2.txt

```
20:53 $ sed 's/\./\.\n/g' breve_historia.txt | sed '/^\s*$/d' > breve_historia_2.txt
✓ ~/Desktop/projects/ssl/utn-ssl-examples/TP1 [feature/tp1|✚ 1]
20:53 $ ls
breve_historia_2.txt  breve_historia.txt  ejemplos_sentencias.sh  script.sh  TP1.pdf
```

d) grep -iF --color 'guerra' breve_historia.txt

```
lemania y Japón en enero de 1944. Igualmente confusos fueron otros actos del nuevo gobier
no. Pero todo ello perdió importancia frente al rápido ascenso de uno de sus miembros, el
 coronel Juan D. Perón, que ocupó la Secretaría de Trabajo y Previsión y, poco después, e
l Ministerio de Guerra. Desde ambos cargos, y gracias a una clara visión de la situación
social del país, Perón pudo construirse una sólida base política.
Reelegido en 1952, su gobierno abarcó desde junio de 1946 hasta septiembre de 1955. Duran
te ese largo plazo, Perón aprovechó la abundante disponibilidad de divisas que el país ha
bía acumulado durante la guerra para financiar una política de abundancia que consolidó s
u posición. Perón aseguró altos salarios a los obreros ,cuyo monto los patronos trasladab
```

e) sed 's/\./\.\n/g' breve_historia.txt | grep --color -E '^A.*(s|s\.)$'

Nota: para este punto se aplico previamente lo del punto a), ya que para el archivo original, "breve_historia.txt", la expresión regular '^A.*(s|s\.)$' no funciona.

```
19:51 $ sed 's/\./\.\n/g' breve_historia.txt | grep --color -E '^A.*(s|s\.)$'
A la crisis del poder central siguió un período en el que cada provincia quedó librada a sus propias fuerzas.
Al subir al poder, en plena guerra mundial, la economía argentina se beneficiaba con la fuerte demanda de materias primas.
```

f) grep --color -c 'peronismo' breve_historia.txt

```
23:13 $ grep --color -c 'peronismo' breve_historia.txt
4
```

g) grep --color -E 'Sarmiento.*Rosas' breve_historia.txt

```
23:15 $ grep --color -E 'Sarmiento.*Rosas' breve_historia.txt
La concepción política de la Federación suponía la unanimidad de opiniones; qui
emigrar, y así se constituyó, sobre todo en Montevideo y en Santiago de Chile,
combatieron al régimen. En Chile escribió Sarmiento el Facundo, vasto intento d
que se escondía detrás del ascenso de Rosas. En Montevideo escribió Echeverría
d, tan próxima a Buenos Aires, los proscriptos procuraron luchar activamente, p
eraban un tirano. Desde allí apoyaron al general Lavalle en sus diversos intent
z cuando las fuerzas adictas a Rosas pusieron sitio a Montevideo en 1843.
```

h) grep --color -E '18[0-9]{2}' breve_historia.txt

```
Desde 1865 hasta 1870, la Argentina mantuvo, junto al Brasil y al Uruguay, una dura guerra
 procuró resolver los problemas internacionales con la mayor equidad.
Para estimular el desarrollo económico, se procuraron capitales extranjeros que se invirti
metros de vías férreas, iniciándose las líneas troncales que nacían en Buenos Aires y conc
contribuyeron al crecimiento de la capital.
Con la definitiva sumisión de los indios y la federalización de Buenos Aires, el president
te importante y que reunía a los grupos influyentes de las provincias. Progresista y liber
ma de gobierno con una fórmula muy significativa: «Paz y administración». Era, sin duda, l
timulaba el optimismo colectivo, en un país que crecía a ojos vistas. Y el gobierno admini
No sin sacudidas, el país aceleró el ritmo de su crecimiento y prosperó visiblemente. Los
antes en el decenio 1890-1899. Había recibido 1.000.000 en el decenio anterior; y en el qu
, que solo contaba con 23.000 habitantes en 1869, llegara a 91.000 en 1895. El país vivía
ial y cultural, menos visible, era la de la formación de un cuerpo social nuevo, de caract
nda, cada una con rasgos sociales y culturales diferentes. En ese mar confuso, los viejos
Pero el desarrollo vertiginoso de la riqueza trajo consigo sus riesgos. El clima de venali
. Muchas fortunas se derrumbaron. Al año siguiente quebraron el Banco Nacional y el de la
```

i) sed 's/^[a-zA-Z]*\b//g' breve_historia.txt

```
 la situación política fue desde el comienzo muy inestable. La sens
de las fuerzas armadas. Muchas figuras del gobierno fueron objetada
os emplazamientos ni las rebeliones abiertas, en tanto que el presi



 aproximarse las elecciones de 1962, las tensiones se acentuaron. E
ón Libertadora. El dilema era, pues, o la proscripción del peronism
dor. Pero los hechos desmintieron esas afirmaciones. Triunfante en
erzas armadas y estas no vacilaron en deponerlo en marzo de 1962.



 vicepresidente José María Guido asumió el poder. Su política fue v
ul» se impuso finalmente y fijó la política del gobierno, que llamó
```

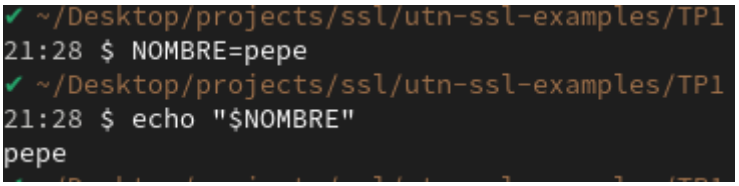
j) ls . | grep --color -nE '.*\.txt'

```
20:21 $ ls . | grep --color -nE '.*\.txt'
1:breve_historia_2.txt
2:breve_historia.txt
```

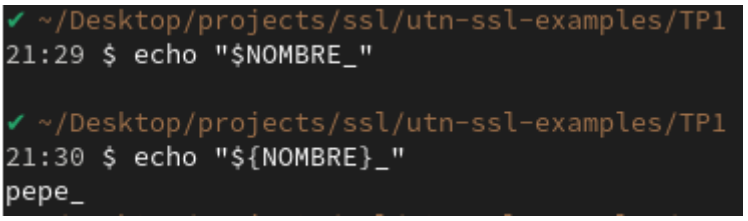
Desarrollo punto 3

- **Variables**

Las variables en Bash se definen como NOMBRE=valor (sin espacios antes o después del símbolo '='). Y su valor se usa, poniendo el símbolo '$' delante del nombre de la variable, $NOMBRE. Si bien, el nombre de las variables no necesariamente tienen que estar en mayúsculas, se las suele utilizar de esa forma.



Si al utilizar el valor de una variable, el nombre de variable es seguido de un carácter que sea otra letra, número o el símbolo '_', hay que agregar los símbolos '{}' alrededor del nombre de la variable.



- **Sentencias condicionales**

Las sentencias condicionales en Bash se definen por **if-then-elif-else** y **case**. En el caso de **if**, para indicar el final de la sentencia se agrega **fi**, y **esac** para el **case**. Ejemplos:

1) **if** (simple): Si se cumple la condición, se ejecuta el echo

```
NOMBRE=Pepe
if [ $NOMBRE == "Pepe"  ]; then
        echo "Es Pepe"
fi
```

2) **if-else**: Si no se cumple la condición del **if**, se ejecuta lo que hay en **else**

```
NOMBRE=Carlos
if [ $NOMBRE == "Pepe" ]; then
        echo "Es Pepe"
else
        echo "No es Pepe"
fi
```

3) **if-elif-else**: Si no se cumple la condición del **if**, se evalúa la condición de **elif**. Y si tampoco se cumple dicha condición, se ejecuta **else**.

```
NOMBRE=Carolina
if [ $NOMBRE == "Pepe" ]; then
        echo "Es Pepe"
elif [ $NOMBRE == "Carlos" ]; then
        echo "Es Carlos"
else
        echo "No es Pepe ni Carlos"
fi
```

Al igual que en otros lenguajes, se pueden anidar tantos **if-elif** como se desee. Pero cuando esto sucede, suele ser más conveniente el uso del **case**.

4) **case**: El valor de la variable se evalúa en los distintos *casos*, y se ejecuta el código correspondiente en aquel que se cumpla la condición.

```
NOMBRE=Andrea
case $NOMBRE in
        "Pepe")
                echo "Es Pepe"
                ;;
        "Carlos")
                echo "Es Carlos"
                ;;
        "Carolina")
                echo "Es Carolina"
                ;;
        *)
                echo "No es un nombre conocido"
                ;;
esac
```

Los '**;;**' indican el fin del bloque. Y la condición por default es representada por '*****', que se ejecuta cuando no se cumple ninguna de las condiciones anteriores. Es por ello que debe ir al final.

- **Sentencias cíclicas**

En este caso se encuentran las sentencias **for** y **while**.
La manera más común de implementar el **for** en Bash, es en su forma **foreach**. Es decir, el **for** itera según la cantidad de elementos de una lista

```
for i in {1..5}
do
        echo "Numero $i"
done
```

En el ejemplo de arriba '{1..5}' representa una lista que contiene los números del 1 al 5, y estos son los valores que va a tomar la variable 'i'.
También se puede ejecutar el **for** al estilo C, es decir con tres expresiones para la inicializacion, condicion e incremento.

```
for (( i=1; i<=5; i++ ))
do
        echo "Numero $i"
done
```

La sentencia while, al igual que en otros lenguajes, itera mientras se cumpla la condición.

```
i=1
while [ $i -le 5 ]
do
        echo "Numero $i"
        i=$((i+1))
done
```

-*le* es una condición que indica “menor o igual que”

- **Subprogramas**

Los subprogramas o funciones tienen la siguiente estructura:

```
mi_funcion() {
        sentencia_1
        sentencia_2
        ....
}
```

Y para invocarla se hace solo por su nombre (sin los paréntesis)

```
mi_funcion
```

Si bien crear una función es bastante simple, no lo es cuando se quiere pasar argumentos a la función y que ésta la utilice, o cuando se quiere utilizar el valor de retorno.

Pasar argumentos a una función

Si queremos pasar, por ejemplo, dos argumentos a una función, se hace de la siguiente manera:

```
mi_funcion "hola" 3
```

Pero para utilizar estos valores desde la función, se tiene que hacer uso de unas ***variables especiales***, las cuales son representadas por $1, $2,…, $9.

Entonces haciendo uso de estas variables, se muestra un ejemplo que simplemente imprime en pantalla lo que recibe la función:

```
mi_funcion() {
        echo "Recibido: $1"
        echo "Recibido: $2"
}


mi_funcion "hola" 3
```

Retornar valores desde una función

En Bash, la palabra ***return*** está reservada para devolver el estado de una función, donde un valor 0 indica que terminó con éxito, y otro valor representa el código de error.

Por lo tanto una forma de utilizar el “valor de retorno”, es mediante variables globales:

```
sumar() {
        SUMA=$(($1+$2))
}

sumar 2 4

echo "La suma es: $SUMA"
```

Por defecto, todas las variables son globales por lo que usarlas puede traer problemas en scripts de gran tamaño.
Otro enfoque que se suele utilizar, es guardar el ***output***, que arroja una función, en una variable. Por ejemplo:

```
sumar_numeros() {
        local resultado=$(($1+$2))
        echo "$resultado"
}

RES=$(sumar_numeros 2 5)

echo "La suma es: $RES"
```

En este ejemplo se utilizó la palabra ***local*** precisamente para hacer que la variable "resultado" solo viva dentro de la función. Luego para guardar el *output* (dado por el comando echo) en una variable en vez de mostrarlo en pantalla, se hace en la línea:

```
RES=$(sumar_numeros 2 5)
```