UNIVERSIDAD POLITÉCNICA DE VICTORIA

UNIDAD II

ADMINISTRACIÓN DE SISTEMAS OPERATIVOS

COMLIZ

August 21, 2018

Supervisor: Dra. Karla E. Vázquez Ortiz

Alumna: Claudia Lizbeth Carrizales Piña

Matrícula: 1730048

UNIVERSIDAD POLITÉCNICA DE VICTORIA
DECLARACIÓN DE AUTORÍA

Yo Claudia Lizbeth Carrizales Piña, declaro que este proyecto titulado: COMLIZ y que el trabajo presentado en este documento son de mi propiedad. Confirmo que:
Este trabajo fue completamente hecho por mi como proyecto de la materia Administración de SO en esta universidad. Que todas las partes de este proyecto fueron sometidas únicamente para la evaluación de esta materia. Este proyecto no ha sido sometido por nadie más. Este proyecto no es copia completa ni parcial de algún otro trabajo.
Este proyecto fue elaborado bajo la supervisión del titular de la materia Administración de SO.

UNIVERSIDAD POLITÉCNICA DE VICTORIA

ABSTRACT: COMLIZ BY CLAUDIA CARRIZALES

En este trabajo utilicé lo que vi en el transcurso de la unidad ya fue lo que es un shell y cómo crear un shell script, utilizar los elementos que lo conforman ya sea meta carácteres, argumentos, comentarios, comandos, entre otros.

Este es un proyecto el cual su propósito es investigar los comandos mas utilizados en linux y posteriormente recrearlos pero en base a una sintaxis propia, creando la estructura desde 0 de nuestro comando mediante expresiones regulares verificar dicha sintaxis de nuestros comandos para después hacer el intercambio y la transformación de nuestra estructura de comando a la estructura predeterminada que debe de tener para poderse ejecutar en shell.

DESARROLLO

Aux.txt guarda línea por línea lo que tiene nuestro archivo de código.txt.
Código.txt es dónde guarda la estructura de los comandos creados por mi.
Ejecutador.sh es dónde se guarda la estructura de codigo.txt pero con la sintaxis ya traducida mediante las expresiones regulares.
Programa.sh sirve como compilador ya que es el encargado de todo el procedimiento para la verificación correcta de la estructura de cada comando hecha por mi.
Prueba.txt es un texto auxiliar para nuetro código.

Programa.sh

```
1  #! /bin/bash
2  echo -e "INICIO DEL PROGRAMA"
3  #Guardamos el total de lineas de nuestro codigo el cual vamos a leer.
4  lineas=$(wc -l Codigo.txt)
5  BandError=0
6  comandos=2
7  #echo "$lineas"
8  #Vamos a quitarle todo lo que no sea un numero a nuestra variable de lineas.
9  lineas=$( echo $lineas | tr -d '[A-Za-z.]')
10 echo -e "NUMERO DE LINEAS DE CODIGO A VERIFICAR $lineas"
11
12 #VERIFICAMOS QUE EN LA PRIMER LINEA ESTE ESCRITO #! /bin/bash
13 LineaPrincipal=$(egrep -o '#!/bin/bash' Codigo.txt)
14 echo "#!/bin/bash" > Ejecutador.sh #VAMOS A RESETEAR NUESTRO DOC DE TEXTO DONDE GUARDAMOS EL CODIGO
   CAMBIADO
15 if [ "$LineaPrincipal" = "#!/bin/bash" ]; then
16     for (( i=2 ; i<=lineas ; i++ ))
17             do
18             GuardarLinea=$(head -$i Codigo.txt| tail -1;)
19             echo "$GuardarLinea" > Aux.txt #le damos a nuestro texto auxiliar lo que guarda la variable
   GuardarLinea
20             #CICLO EL CUAL BUSCA LA PALABRA CLAVE DE ALGUN COMANDO Y POSTERIORMENTE DE ENCONTRARLA ROMPE
   EL CICLO
21             for (( j=0 ; j<comandos ; j++ ))
22                 do
```

Primeramente lo que realicé es guardar la cantidad de líneas de nuestro codigo.txt cómo se muestra en la línea 4, se inicializa una variable en este caso "BandError" para cuando marque errores de síntaxis, después a nuestra variable lineas con tr le quitamos todas las letras y punto ya que al momento de guardar la cantidad de líneas se guarda también de donde viene y a nosotros sólo nos interesa saber la cantidad de líneas.
Después procedemos a ver que esté el bin bash si esto no se cumple entonces no entrará a nuestro ciclo y marcará error.
Al entrar en nuestra condición cuando es verdadera entonces lo que se hace es entrar a un ciclo desde i=2 hasta que sea menor o igual que el tamaño de nuestras íneas, empieza en 2 porque ya la primer linea se verificó con el bin bash.
Después guardamos en nuestra variable "GuardarLineas" usamos la función tail la cuál nos permite guardar línea por línea de nuestro codigo.txt.
Después mandamos lo que almacena nuestra variable a nuestro texto Aux.txt el cuál almacenará la línea y es desde ese texto que haremos la verificación.

Programa.sh

```
GuardarLinea
            #CICLO EL CUAL BUSCA LA PALABRA CLAVE DE ALGUN COMANDO Y POSTERIORMENTE DE ENCONTRARLA ROMPE
EL CICLO
            for (( j=0 ; j<comandos ; j++ ))
                do
                    GuardarLinea=""
                    #VERIFICAR SI SE INICIALIZO UNA VARIABLE
                    GuardarLinea=$(egrep -o '\<crear\>' Aux.txt)
                    if [ "$GuardarLinea" = "crear" ]; then
                        break
                    fi

                    #COMANDO ECHO
                    GuardarLinea=$(egrep -o '\<mostrar\>' Aux.txt)
                    if [ "$GuardarLinea" = "mostrar" ]; then
                        break
                    fi
                    #FIN DE COMANDO ECHO

                    #VERIFICACION COMANDO IF
                    GuardarLinea=$(egrep -o '\<yes\>' Aux.txt)
                    if [ "$GuardarLinea" = "yes" ]; then
                        break
                    fi
                    GuardarLinea=$(egrep -o '\<yesno\>' Aux.txt)
```

sh Ancho de la tabulación: 4 Ln 1, Col 1 INS

Luego tengo un ciclo el cuál va a ir avanzando y comparando lo que tiene nuestra variable que almacena la línea hasta que encuentre coincidencia con una palabra clave de nuestros comando, cuando esto suceda entonces va a romper el ciclo.

Programa.sh ×

```
124             #UTILIZAMOS UN CASE EL CUAL SIRVE PARA SABER DE CUAL COMANDO VAMOS A VERIFICAR SU SINTAXIS Y
    POSTERIORMENTE CONVERTIRLO
125             case $GuardarLinea in
126                 "mostrar")
127                     GuardarLinea=$(egrep -o 'mostrar +\([[:alnum:][:punct:][:space:]]+\)$' Aux.txt)
128                     if [ "$GuardarLinea" != "" ]; then #si nuestra variable tiene algo almacenado
    significa que si encontro el patron de texto buscado
129                     #utilizamos sed -i para cambiar nuestro texto a como deberia de ir normalmente
130                         sed -i 's/mostrar/echo/g' "Aux.txt"
131                         sed -i 's/(/"/g' "Aux.txt"
132                         sed -i 's/)/"/g' "Aux.txt"
133                         arrayEjecutador[i]=$(cat Aux.txt)
134                         echo "${arrayEjecutador[i]}" >> Ejecutador.sh
135                     else
136                         echo "Error Linea $i"
137                         BandError=1 #enciendo la bandera para marcar error
138                         i=`expr $lineas + 1` #si hubo un error entonces forzo a cerrar nuestro ciclo
139                     fi
140                     ;;
141                 "yes")
142                     #COMANDO IF | YES
143                     #GuardarLinea=$(head -$i Codigo.txt| tail -1;)
144                     #echo "$GuardarLinea" > Aux.txt
145                     GuardarLinea=$(egrep -o 'yes+[[:space:]]+{+[[:space:]]+[[:alnum:][:space:][:punct:]]
    +[[:space:]]+}+;+[[:space:]]+do$' Aux.txt)
```

Después mediante un case dependiendo de la palabra clave que haya encontrado referente a un comando va a entrar y va a sobrescribirse en la misma variable mediante egrep va a guardarse el patrón buscado, mediante una expresión regular, si nuestra variable es diferente a vacio esto quiere decir que el patrón (sintaxis) es correcta y procederemos a ir cambiando lo necesario mediente sed -i para convertirlo en la estructura correspondiente a la sintaxis normal de shell, después guardamos en un array lo que contiene aux.txt con cat de esta manera ya almacenamos en nuestro array la línea de comando ya estructurada correctamente y después mandamos esa misma línea a nuestro Ejecutador.sh.

Programa.sh ×

```
145                     GuardarLinea=$(egrep -o 'yes+[[:space:]]+{+[[:space:]]+[[:alnum:][:space:]][:punct
    +[[:space:]]+}+;+[[:space:]]+do$' Aux.txt)
146                     if [ "$GuardarLinea" != "" ]; then
147                         sed -i 's/yes/if/g' "Aux.txt"
148                         sed -i 's/do/then/g' "Aux.txt"
149                         sed -i 's/{/[/g' "Aux.txt"
150                         sed -i 's/}/]/g' "Aux.txt"
151                         arrayEjecutador[i]=$(cat Aux.txt)
152                         echo "${arrayEjecutador[i]}" >> Ejecutador.sh
153                     else
154                         echo "Error Linea $i"
155                         BandError=1 #enciendo la bandera para marcar error
156                         i=`expr $lineas + 1` #si hubo un error entonces forzo a cerrar nuestro ciclo
157                     fi
158                     ;;
159                 "yesno")
160                     sed -i 's/yesno/else/g' "Aux.txt"
161                     arrayEjecutador[i]=$(cat Aux.txt)
162                     echo "${arrayEjecutador[i]}" >> Ejecutador.sh
163                     ;;
164                 "sey")
165                     sed -i 's/sey/fi/g' "Aux.txt"
166                     arrayEjecutador[i]=$(cat Aux.txt)
167                     echo "${arrayEjecutador[i]}" >> Ejecutador.sh
168                     ;;
169                 "mientras")
```

Si por lo contrario nuestra variable esta vacía esto quiere decir que si encontró la palabra clave, pero el patrón (la sintaxis) no es la esperada y se enciende nuestra bandera de error y forzamos el cierre del barrido de nuestras líneas del texto ya que no nos interesa seguir verificando líneas si ya encontramos un error.

Codigo.txt ×

```
#!/bin/bash
mostrar (hola)
crear y=4
mostrar ($y)
&Este es un comentario
yes { 5 -lt 8 }; do
    mostrar (funciona el if)
yesno
    mostrar (funciona else)
sey

mientras { 5 -lt 8 }; hacer
    mostrar (entraste al while)
    romper
sartneim

extraeInfo Prueba.txt
dia
operacion 4 + 5
factorial 20

mostrar (Ingresa un valor)
leer x
mostrar (El valor ingresado es: $x)
```

Texto pla

Estructura de los comandos creados por mi, la sintaxis propuesta.

Ejecutador.sh

```
#!/bin/bash
 y=4
echo "$y"
#Este es un comentario
if [ 5 -lt 8 ]; then
    echo "funciona el if"
else
    echo "funciona else"
fi
while [ 5 -lt 8 ]; do
    echo "entraste al while"
    break
done
cat Prueba.txt
date
expr 4 + 5
factor 20
echo "Ingresa un valor"
read x
echo "El valor ingresado es: $x"
for (( i=1 ; i<=x ; i++ ))
do
echo "Entro jeje "
done
```

Muestra nuestro código ya cambiado a la estructura (sintaxis) correspondiente.

CONCLUSIÓN

Pude lograr el objetivo investigando mucho unos comando que me ayudaron a realizar esto y utilizando las diapositivas de la unidad, y al salir el primer comando ya se fácilitaron los demás dicho esto con diferente estructura pero se logró el objetivo que tenía en mente se logró realizar los comandos con nuestra sintaxis propuesta ya es el caso del while, si entonces, for, escribir, leer, date, factor, exp, cat.