2019

Proyecto

CIFRADO CÉSAR

ANDRÈS AMADEUS GALLARDO TINOCO

Variables Globales

```
}
#Variables globales como alfabeto y numero de desplazamiento
estado=0
declare -a alfabeto=(a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z)
letras=$(echo ${#alfabeto[@]})
pasos=4
codigo=""
tipo=""
#Menu principal (lectura de banderas)
```

Son seis variables locales las cuales guardan:

- Estado: es el tipo de funcion que va a realizar cifrar, descifrar o ejecutar
- Alfabeto: arreglo que contiene el alfabeto sobre el cual se va a trabajar
- Letras: numero de letras que contiene el arreglo alfabeto
- Pasos: es el número de desplazamiento que va a realizar (cantidad de pasos desplazados)
- Codigo: nombre del archivo sobre el cual se va a trabajar
- Tipo: bandera que indica si se cifra o descifra el codigo

Menú principal (lectura de banderas)

```
#Menu principal (lectura de banderas)
while getopts ":e:d:s:a:z:" op
do
    case $op in
        e)
            codigo=$OPTARG
            codigotemp=$(echo temporal)
            cat $OPTARG | sed 's/\t/®®®®/g' | sed 's/ /®/g' >  $(echo temporal)
            estado=1
        ;;
        d)
            codigo=$OPTARG
            codigotemp=$(echo temporal)
            cp $OPTARG $(echo temporal)
            estado=2
        ;;
        s)
            pasos=$OPTARG
        ;;
        a)
            abctemp=$OPTARG
            unset alfabeto
            alfabeto=()
            for i in $(seq 0 $((${#abctemp} - 1)));
            do
                alfabeto[i]=${abctemp:$i:1}
            done
            letras=$(echo ${#alfabeto[@]})
        ;;
        z)
            codigo=$OPTARG
            codigotemp=$(echo temporal)
            codigotemp2=$(echo temp2$codigo)
            cat $OPTARG | sed 's/\t/®®®®/g' | sed 's/ /®/g' >  $(echo temporal)
            cat $OPTARG >  $(echo temp2$codigo)
            estado=3
        ;;
    esac
done
#Ejecucion de estados
```

Getopts: funciona para recibir banderas, que son los atributos de cuando se ejecuta el codigo.

Posibles casos:

• E: encripta el archivo, para ello primero lee ($OPTARGS) el cual es el argumento despues de la bandera y lo almacena en codigo (variable global) y asi mismo crea un archivo temporal para manejar el cifrado y lo guarda (el nombre) en otra variable y coloca el estado en 1.
Anteriormente cuando crea el archivo copia todo el contenido del original en el temporal usando 2 filtros, el primero cambia los espacios y el segundo las tabulaciones por 4 símbolos de espacio.

- D: realiza lo mismo que E pero sin los filtros y coloca el estado en 2
- S: modifica la variable pasos por el argumento seguido la bandera

- A: modifica la variable alfabeto y despues recalcula el tamaño de este y lo guarda en la variable local letras
- Z: crea 2 archivos temporales para poder obtener el codigo, descifrarlo y ejecutarlo en otro.

Estados

```
case $estado in
    1)
        tipo="c"
        true>$codigo
        echo " " >> $codigotemp
        while read  line;
        do
            contenido=$line
            codificacion
            echo $contenido >> $codigo
        done < $codigotemp
        rm $codigotemp
    ;;
    2)
        tipo="d"
        true>$codigo
        while read line;
        do
            contenido=$line
            codificacion
            echo $contenido | sed 's/®/ /g' >> $codigo
        done < $codigotemp
        rm $codigotemp
    ;;
    3)
        tipo="d"
        true>$codigotemp2
        while read line;
        do
            contenido=$line
            codificacion
            echo $contenido | sed 's/®/ /g' >> $codigotemp2
        done < $codigotemp
        rm $codigotemp
        . $codigotemp2
        rm $codigotemp2
    ;;
esac
```

Son 3 estados cifrar, descifrar y ejecutar.

- 1 (cifrar): coloca la bandera global tipo en c (Cifrar) e indica que al archivo original (previamente guardado su nombre en la variable global codigo) se sobrescribe, concatena un salto de linea al final del codigo y comienza un while que obtendrá linea por linea del archivo temporal (copia del archivo original) lo guardara en contenido, lo mandara a cifrar y lo concatenara en el archivo original, una vez terminado este proceso borrara el archivo temporal.
- 2 (descifrar): Exactamente lo mismo que el estado 1 solo que la bandera tipo se coloca en d (Descifrar) y cuando se concatena el texto ya descifrado se manda por un filtro que recupera los espacios (identación)
- 3 (Ejecución): coloca la bandera tipo en d (Descifrar) y junto con 2 archivos temporales realiza su tarea, el primero guarda el codigo cifrado mientras que el segundo ira almacenando el codigo descifrado y lo ejecutara al final, una vez terminado el proceso se eliminaran ambos archivos.

Módulos

```
#Funciones auxiliares para el encriptado y Desencri
function modulo {
    cadena=()
    for i in $(seq 0 $((${#contenido} - 1)));
    do
        cadena[i]=${contenido:$i:1}
    done
    lcadena=$(echo ${#cadena[@]})
    temp1=0
}
function modulo2 {
    contenido=""
    for i in ${cadena[*]}
    do
        contenido=$(echo $contenido$i)
    done
}
function cifrar {
```

Ambos módulos tienen la misma funcion que es pasar una cadena a un arreglo o viceversa, esta cadena es la que se tiene que cifrar, la diferencia aparte de la anterior es que modulo crea una variable y obtiene el tamaño de la cadena.

```
}
function cifrar {
    a=$(($temp + $pasos))
    if [ $a -ge $(($letras)) ];
    then
        while [ $a -ge $(($letras)) ]
        do
            a=$(($temp + $pasos - $letras))
        done
    else
        a=$(($temp + $pasos))
    fi
}
function descifrar {
    a=$(($temp - $pasos))
    if [ $a -lt 0 ];
    then
        while [ $a -lt 0 ]
        do
            a=$(($temp - $pasos + $letras))
        done
    else
        a=$(($temp - $pasos))
    fi
}
function codificacion {
```

Las funciones cifrar al igual que la funcion descifrar esta formado por 2 secciones la primera es un if que compara el tamaño del desplazamiento y la segunda se encarga de corregir este anterior.

Primero ingresa al if y con un while corrige el tamaño del desplazamiento, por ejemplo si el alfabeto es de tamaño 30 y el desplazamiento es de 72, este realiza un modulo 30 sobre este (funcion while) y despues con el residuo realiza el desplazamiento, asi mismo funciona descifrar solo que uno suma el numero de pasos y el otro los resta.

```
}
function codificacion {
    modulo
    for (( i=0; i<lcadena; i++))
    do
        encontrado=`echo ${alfabeto[*]} | grep "${cadena[i]}"`
        if [ "${encontrado}" != "" ];
        then
            temp=0
            for j in ${alfabeto[*]}
            do
                if [ $j == "${cadena[$i]}" ]
                then
                    if [ "$tipo" == "c" ]
                    then
                        cifrar
                    else
                        descifrar
                    fi
                    cadena[$temp1]=${alfabeto[$a]}
                    break
                fi
                temp=$(($temp + 1))
            done
        else
            esta=0
        fi
        temp1=$(($temp1 + 1))
    done
    modulo2a
}
#Variables globales como alfabeto y numero de desplazamiento
```

Codificación

Esta funcion es la principal y empieza usando el modulo para pasar la cadena recibida a un arreglo y a partir de ahí recorre el arreglo con un for, despues verifica que ese carácter se encuentre en el alfabeto, el siguiente for busca la posición del carácter dentro del alfabeto y por último le suma a la letra el desplazamiento a partir del índice encontrado.

Dentro del ultimo if se encuentran los módulos cifrar y descifrar y a partir de estos y sus bandera realizaran el desplazamiento.

Referencias:

https://baulderasec.wordpress.com/desde-la-consola/shell-en-unixlinux-sh-ksh-bash/4-bases-de-la-programacion-shell/5-el-comando-read/

https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar

https://enavas.blogspot.com/2008/03/el-shell-de-linux-comando-tr.html

https://blog.carreralinux.com.ar/2016/09/convertir-tabulaciones-en-espacios-viceversa/

https://www.linuxquestions.org/questions/linux-newbie-8/tab-in-bash-script-242400/

https://www.ibm.com/developerworks/ssa/linux/library/l-lpic1-v3-103-2/index.html