

```

1  12a)
2  #!/bin/bash
3  #solicite al usuario 2 numeros, imprima la
4  suma,resta,multiplicacion, y cual
5  #es el mayor de los dos
6  echo -n "Ingresa primer numero: "
7  read n1
8  echo -n "Ingresa segundo numero: "
9  read n2
10 echo -n "La suma de $n1 mas $n2 es: "
11 echo `expr $n1 + $n2`
12 echo -n "La resta de $n1 menos $n2 es: "
13 echo `expr $n1 - $n2`
14 echo -n "La multiplicacion entre $n1 por $n2 es: "
15 echo `expr $n1 '*' $n2`
16 if [ $n1 -lt $n2 ]
17 then
18     echo "El numero mayor es $n2"
19 else
20     echo "El numero mayor es $n1"
21 fi
22
23 12b)
24 #!/bin/bash
25 # 2 numeros como parametro, imprima la
26 suma,resta,multiplicacion, y cual
27 #es el mayor de los dos
28 if [ $# -ne 0 ]
29 then
30     echo -n "La suma de $1 mas $2 es: "
31     echo `expr $1 + $2`
32     echo -n "La resta de $1 menos $2 es: "
33     echo `expr $1 - $2`
34     echo -n "La multiplicacion entre $1 por $2 es: "
35     echo `expr $1 '*' $2`
36     if [ $1 -lt $2 ]
37     then
38         echo "El numero mayor es $2"
39     else
40         echo "El numero mayor es $1"
41     fi
42 else
43     echo "No pasaste ningun argumento"
44 fi
45
46 13a)

```

```

47 #!/bin/bash
48 #visualizar en pantalla los numeros del 1 al 100 asi
49 como sus cuadrados
50 for ((i=0;i<=100;i++))
51 do
52     echo "Numero $i y su cuadrado `expr $i '*' $i`"
53 done
54
55 13b)
56 #!/bin/bash
57 #mostrar 3 opciones, segun la elegida listo, pwd, y
58 quien
59 echo "1) Listar el contenido del directorio actual"
60 echo "2) El nombre del directorio actual"
61 echo "3) Quien esta logeado al sistema"
62 echo -ne "Ingresa numero de opcion: "
63 read opcion
64 case $opcion in
65     1)
66         echo "El contenido del directorio
67 actual es `ls -l`"
68         ;;
69     2 )
70         echo "La ruta del directorio actual es
71 `pwd`"
72         ;;
73     3)
74         echo "Estas logeado al sistema como
75 `whoami`"
76         ;;
77     *)
78         echo "Opcion incorrecta"
79         ;;
80 esac
81
82 13c)
83 #!/bin/bash
84 #recibe como parametro un nombre de archivo e
85 informa si existe o no.
86 #si existe muestra si es directorio o archivo
87 #si no existe el archivo/directorio crea un directorio
88 con el nombre recibido como parametro
89 if [ $# -ne 0 ]
90 then
91     if [ -f $1 ]
92     then

```

```

93         echo "Existe y es un archivo"
94     elif [ -d $1 ]
95     then
96         echo "Existe y es un directorio"
97     else
98         echo "No existe archivo/directorio.
99 Creando directorio $1"
100     fi
101
102 else
103     echo "Debe pasar un nombre de
104 archivo/directorio como parametro"
105 fi
106
107 14)
108 #!/bin/bash
109 #dados dos vectores de longitud igual, pero no se
110 conocen, sumar elemento a elemento e imprimir
111 vector1=(1 15 9 20)
112 vector2=(3 1 10 40)
113
114 for((i=0; i<${#vector1[@]}; i++))
115 do
116     echo "La suma de los elementos de la posicion
117 $i es : `expr ${vector1[$i]} + ${vector2[$i]}`"
118 done
119
120 14)
121 #!/bin/bash
122 #renombrar archivos, recibe el directorio pasado como
123 parametro
124 #-a CADENA al final
125 #-b CADENA al principio
126 recorrer ()
127 {
128     cd $1
129     for i in `ls`
130     do
131         if [ -f "$i" ]
132         then
133             renombrar $2 $3 "$i"
134         fi
135     done
136 done
137
138 }

```

```

139
140 renombrar()
141 {
142     if [ $1 = "-a" ]
143     then
144         mv $3 ${3}${2}
145     else
146         mv $3 ${2}${3}
147     fi
148 }
149
150 if [ $# -eq 3 ]
151 then
152     recorrer $1 $2 $3
153 else
154     echo "Cantidad incorrecta de parametros"
155 fi
156
157 15)
158 #!/bin/bash
159 #cut
160 #ls -l | cut -d " " -f 1,2 muestra columnas 1 y 2
161 delimitados por " "
162 #ls -l | tr -s " " | cut -d " " -f 1,2 el tr me elimina " "
163 repetidos y me deja solo uno
164     echo `man cut`
165
166 16)
167 #!/bin/bash
168 #recibo por parametro una extension, y hace un
169 reporte con 2 columnas, el nombre de user y la
170 cantidad de archivos
171 #que posee con esa extension. Se guarda en un archivo
172 llamado reporte.txt
173 #ls -l | cut -d " " -f 3 -> se queda con la columna de
174 usuario tras un ls -l
175 #awk '{print $3,$4}' ls.txt imprime la columna 3 y 4 del
176 archivo ls.txt
177
178 #`ls -l > ls.txt`
179     #echo `awk '{print $3}' ls.txt | wc -l`
180     #echo "$3"
181
182 #ls -l | cut -d " " -f 3 -> se queda con la columna de
183 usuario tras un ls -l

```

```

184 #ls -l | tr -s " " | cut -d " " -f 3 te elimina los doble
185 espacio " "
186 #ls -l | grep '.sh' | wc -l me quedo con la cantidad de
187 lineas .sh
188
189 if [ $# -eq 1 ]
190 then
191     path=`pwd`
192     string=
193     cantidad=
194     cd /home
195     for user in `ls -l | tr -s " " | cut -d " " -f 3`
196     do
197         cd /home
198         cd $user
199         string=$user
200         cantidad=`ls -l | grep $1 | wc -l`
201         echo $string" "$cantidad >>
202     $path"/reporte.txt"
203     done
204 else
205     echo "No pasaste ningun parametro"
206 fi
207
208 18)
209 #!/bin/bash
210 #verificar cada 10 seg si un usuario se ha logueado en
211 el sistema, el nombre viene por parametro
212 #cuando se loguea debe mostrar "Usuario $1 logueado
213 en el sistema"
214
215 control()
216 {
217     for user in `who | cut -d " " -f1`
218     do
219         if [ $user == $1 ]
220         then
221             echo "Usuario $1 logueado
222 en el sistema"
223             exit
224         fi
225     done
226
227 }
228
229 if [ $# -eq 1 ]

```

```

230 then
231     control $1
232     while [ 0 -lt 1 ]
233     do
234         sleep 10; control $1
235     done
236 else
237     echo "Debe pasar un nombre de usuario"
238
239 fi
240
241 19)
242 #!/bin/bash
243 #muestra un menu con la selecciona de cada uno de los
244 scripts creados.
245 #el menu debe permanecer activo hasta que se
246 precione salir
247
248 while [ 0 -ne 1 ]
249 do
250     echo "MENU DE COMANDOS"
251     echo "01. Script 1"
252     echo "00. Salir"
253     echo -ne "Ingrese numero de opcion a
254 ejecutar: "
255     read opcion
256     case $opcion in
257         01)
258             echo "eligio la opcion 01"
259             ;;
260
261         00)
262             exit
263             ;;
264
265         *)
266             ;;
267     esac
268 done
269
270 20)
271 #!/bin/bash
272 #el comportamiento de una pila
273 #21) Dentro del mismo script y utilizando las funciones
274 implementadas agregue 10

```

```

275 #elementos a la pila, saque 3 de ellos, imprima la
276 longitud de la cola y luego la totalidad
277 #de los elementos que en ella se encuentran.
278 vector=()
279 index=0
280
281 print()
282 {
283     for elem in ${vector[*]}
284     do
285         echo $elem
286     done
287 }
288
289 function length
290 {
291     return `expr $index + 1`
292 }
293
294 pop()
295 {
296     unset vector[index]
297     let index--
298 }
299
300 push()
301 {
302     vector[index]=$1
303     let index++
304 }
305
306 push 1
307 push 2
308 push 3
309 push 4
310 push 5
311 push 6
312 push 7
313 push 8
314 push 9
315 push 0
316 #el primer pop no hace nada
317 pop
318 pop
319 pop
320 pop

```

```

321 print
322 #imprimir valor de funcion
323
324 22)
325 #!/bin/bash
326 #dado el array multiplicarlo atravez de una funcuion
327 num=(11 3 5 7 9 3 5 4)
328 productoria()
329 {
330     resultado=1
331     for x in ${num[*]}
332     do
333         #echo $x
334         resultado=`expr ${resultado} '*' ${x}`
335     done
336     echo "El resultado de la multiplicacion es:
337 $resultado"
338 }
339 }
340
341 productoria ${num[*]}
342 23)
343 #!/bin/bash
344 #recorre un arreglo e imprime los numeros pares, y
345 cuenta los numeros impares
346
347 vector=(1 2 3 4 5 6 7 8 9 10)
348
349 impares=
350
351 for elem in ${vector[@]}
352 do
353     mod=`expr $elem % 2`
354     if [ $mod -eq 0 ]
355     then
356         echo $elem
357     else
358         let impares++
359     fi
360 done
361 echo "La cantidad de numeros impares es $impares"
362
363 #!/bin/bash
364 #dados dos vectores de longitud igual, pero no se
365 conocen, sumar elemento a elemento e imprimir
366

```

```

367 vector1=(1 15 9 20)
368 vector2=(3 1 10 40)
369
370 for((i=0; i<${#vector1[@]}; i++))
371 do
372     echo "La suma de los elementos de la posicion
373 $i es : `expr ${vector1[$i]} + ${vector2[$i]}`"
374 done
375
376 Calculadora)
377 #!/bin/bash
378 #calculadora, recibe la operacion y los numeros por
379 parametro
380 #ejemplo: 2 + 2
381
382 if [ $# -eq 3 ]
383 then
384     case $2 in
385         +)
386             echo "La suma entre $1 y $3 es `expr
387 $1 + $3`"
388             ;;
389         -)
390             echo "La resta entre $1 y $3 es `expr
391 $1 - $3`"
392             ;;
393         '*' )
394             echo "La multiplicacion entre $1 y $3
395 es `expr $1 '*' $3`"
396             ;;
397         %)
398             echo "La division entre $1 y $3 es
399 `expr $1 / $3`"
400             ;;
401         *)
402             ;;
403     esac
404 else
405     echo "Numero de parametros incorrectos.
406 Ejemplo: 2 + 2"
407 fi
408
409 ASCII)
410 #!/bin/bash
411 #identificar archivos ASCII dentro de los home de
412 usuario

```

```

413
414 virus=0
415 cd /home
416 for user in `ls`
417 do
418     cd /home
419     cd $user
420     for archivo in `ls | grep ".txt"`
421     do
422         if [ `file $archivo | grep ASCII` == ]
423         then
424             echo "Es un virus $archivo"
425             let virus++
426         fi
427     done
428 done
429 echo "La cantidad de archivos infectados eliminados es:
430 $virus"
431
432 funcion)
433 #!/bin/bash
434 # imprimir lo que devuelve una funcion
435
436 function funcion
437 {
438     return 1
439 }
440
441 funcion
442 echo $?
443
444 mover)
445 #!/bin/bash
446 #implemente el comando mover. Recibe dos
447 parametros, el primero el origen y el segundo el
448 destino
449 #debera mantener un archivo de log en /var/log/mover
450 donde por cada invocacion del script
451 #debera escribir una linea que contenga: "El archivo
452 <nombre> ha sido movido a <destino>"
453
454
455 if [ $# -eq 2 ]
456 then
457     if [ -e $1 ]
458     then

```

```
459             mv $1 $2
460             echo "El archivo $1 ha sido movido a
461 $2" >> /home/frank/Escritorio/mover
462             exit 0
463         else
464             echo No existe el archivo/directorio
465 origen
466             exit 1
467         fi
468
469     else
470         echo Numero de parametros incorrectos
471         exit 1
472     fi
473
474 buscar)
475 #!/bin/bash
476 #dar dos opciones, ingrese el patron a buscar y el
477 nombre del archivo destino
478 #busca en todo /etc los archivos que concuerden con
479 $1 y los empaquete y comprime en un unico archivo
480 #con nombre de $2
481
482 echo -n "Ingrese patron a buscar: "
483 read patron
484 echo -n "Ingrese el nombre del archivo destino: "
485 read archivoDestino
486 cd /etc
487 tar -cvzf $archivoDestino `ls | grep $patron`
```