Comenzado el	domingo, 9 de noviembre de 2025, 10:11
Estado	Finalizado
Finalizado en	domingo, 9 de noviembre de 2025, 11:45
Tiempo empleado	1 hora 33 minutos
Calificación	**19,00** de 20,00 (**95**%)

Pregunta 1

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué hace la siguiente secuencia de comandos?

```
tar -czvf zip.tar.gz $(ls); tar -xvfz zip.tar.gz -C /tmp
```

a. Genera un archivo comprimido con el contenido del directorio actual y lo descomprime en /tmp ✔

b. Genera un archivo empaquetado con el contenido del directorio actual.

c. Genera un archivo comprimido con el contenido del directorio actual y lo mueve al directorio /tmp

d. Falla, no se le da una lista de archivos

Pregunta 2

Correcta

Se puntúa 1,00 sobre 1,00

¿Que resultado tiene el siguiente comando "cat /etc/passwd | cut -f1 -d: | grep "^a"" ?

a. Imprime los nombre de usuario que contenga una letra a

b. Imprime los nombres de los usuarios que empiecen con la letra a ✔

c. Imprime los homes de los usuarios que tienen una letra a

d. No imprime nada

e. Imprime las password de los usuarios que contenga una letra a

Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuales de los siguientes usos de variables son correctos?

- [x] a. echo $NOMBRE ✔
- [ ] b. APELLIDO = "sanchez"
- [ ] c. echo APELLIDO
- [x] d. DIRECCION="56 nro 436" ✔
- [x] e. echo ${DIRECCION} ✔
- [ ] f. var NOMBRE="pepe"

Pregunta 4

Correcta

Se puntúa 1,00 sobre 1,00

¿Cualés de las siguientes son funciones correctamente definidas?

- [ ] a.
```
function x(1, 2){
    echo $1 | grep $2
}
```
- [x] b.
```
function x{
    echo $1 | grep $2
}
```
✔
- [ ] c.
```
function x(1, 2); do
    echo $1 | grep $2
done
```
- [ ] d.
```
x(1, 2){
    echo $1 | grep $2
}
```
- [x] e.
```
x(){
    echo $1 | grep $2
}
```
✔

Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

Se desea hacer un script que imprima la lista de argumentos que recibe. ¿Cuales de las siguientes implementaciones son correctas?.

a.
```
#!/bin/sh
for arg in $*; do
    echo $arg
done
```
✔

b.
```
#!/bin/sh
echo $*
```
✔

c.
```
#!/bin/sh
for arg in $@; do
    echo $arg
done
```
✔

d.
```
#!/bin/sh
for arg in ${argv[*]}; do
    echo $arg
done
```

e.
```
#!/bin/sh
echo $#
```

f.
```
#!/bin/sh
for arg in argv; do
    echo $arg
done
```

Pregunta 6

Correcta

Se puntúa 1,00 sobre 1,00

Supongamos que tenemos un archivo llamado "archivo" con el siguiente contenido:

```
eth2 Link encap:Ethernet HWaddr 00:e0:7d:b4:1c:38
inet addr:192.168.1.112 Bcast:255.255.255.255
Mask:255.255.255.0
inet6 addr: fe80::2e0:7dff:feb4:1c38/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:115588267 errors:0 dropped:0 overruns:0 frame:0
TX packets:35601221 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:3240134640 (3.2 GB) TX bytes:2042178372 (2.0 GB)
Interrupt:20 Base address:0xce00
```

que resultado se obtiene al ejecutar:

cat archivo | grep inet | cut -d: -f2 | cut -d" " -f1

- a. 192.168.1.112 ✔
- b. fe80::2e0:7dff:feb4:1c38/64
- c. Ningun resultado
- d. 00:e0:7d:b4:1c:38
- e. 255.255.255.255

Pregunta 7

Incorrecta

Se puntúa 0,00 sobre 1,00

Dado el siguiente script, ¿Que afirmaciones son verdaderas acerca de su ejecucion?

```
#!/bin/bash
for i in {1..100}; do
 while true; do
  if ! (($i % 25)); then
   echo "$i es divisible por 25"
   continue 2
  elif [ $i -eq 53 ]; then
   break 2
  elif [ "$i % $i" ]; then
   break
  fi
 done
done
```

- a. Es un bucle infinito ✖
- b. Cuando el valor de i llega al 53, el script termina.
- c. Imprime que los valores 25 y 50 son divisibles por 25
- d. Imprime todos los valores del 1 al 100 divisibles por 25

Pregunta 8

Correcta

Se puntúa 1,00 sobre 1,00

Se desea hacer un script que imprima su primer argumento en pantalla, por ejemplo:

$ ./mi_script.sh "Hola Mundo"

debe imprimir "Hola Mundo" (sin las comillas).

¿Cuál de las siguientes es una implementación que cumple este objetivo?

a.
```
#!/bin/sh
echo $argv[1]
```

b.
```
#!/bin/sh
echo ${argv[0]}
```

c. ✔
```
#!/bin/sh
echo $1
```

d. ✔
```
#!/bin/sh
for i; do
echo $i
break
done
```

e.
```
#!/bin/sh
echo argv[0]
```

f.
```
#!/bin/sh
echo argv[1]
```

g.
```
#!/bin/sh
echo $0
```

h.
```
#!/bin/sh
echo ${argv[1]}
```

Pregunta 9

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuales de los siguientes usos de arreglos son correctos?

a. echo ${arreglo[1]} ✔
b. echo $arreglo[2]
c. arreglo = (3 4 5)
d. echo ${#arreglo[*]} ✔
e. arreglo=(3 4 5) ✔
f. agregar un nuevo elemento: arreglo+=5
g. agregar un nuevo elemento: arreglo+=(5) ✔

Pregunta 10

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuales de los siguientes usos de expr es correcto?

- [x] a. expr 5 != 5 ✔
- [x] b. expr length "PEPE" ✔
- [ ] c. expr 4 5 *
- [x] d. expr 4 + 5 ✔
- [ ] e. expr - 6 5

Pregunta 11

Correcta

Se puntúa 1,00 sobre 1,00

¿Que hace el comando test -w /home/pepe/ ?

- [ ] a. Evalua si el archivo o directorio existe y es un archivo regular
- [ ] b. Evalua si el archivo o directorio existe y se tiene el permiso de lectura
- [ ] c. Evalua si el archivo o directorio existe y es un directorio
- [x] d. Evalua si el archivo o directorio existe y se tiene el permiso de escritura ✔
- [ ] e. Evalua si el archivo o directorio existe

Pregunta 12

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuales de las siguientes sintaxis del comando if son correctas?

- [x] a. ✔

```
if [ "$nombre" == "Maria" ]
then
echo "Es igual"
fi
```

- [x] b. if test -r /home/pepe; then echo "Tengo permisos de lectura"; fi ✔
- [ ] c. if test -d /home/pepe then echo "es un directorio" fi
- [ ] d.

```
if ["$nombre" == "Maria"]
then echo "Es igual"
fi
```

Pregunta 13

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuales de las siguientes sintaxis de funciones es correcta?

a.
```
mayor(a,b) {
if [ $a > $b ]; then echo $a; else echo $b; fi
}
```

b.
```
funtion mayor() {
if [ $1 > $2 ]; then echo $1; else echo $2; fi
}
```

c. ✔
```
mayor() {
if [ $1 > $2 ]; then echo $1; else echo $2; fi
}
```

Pregunta 14

Correcta

Se puntúa 1,00 sobre 1,00

Qué opciones son verdaderas respecto a la secuencia de comandos:

```
(test -f archivo && grep menta archivo && echo Z) || echo Q
```

- [x] a. Si "archivo" existe y contiene el string "menta" imprime "Z", sino imprime "Q" ✔
- [ ] b. Sin los paréntesis el resultado sería distinto
- [ ] c. Si "archivo" existe y contiene el string "menta" imprime "Q", sino imprime "Z"
- [x] d. Sin los paréntesis el resultado sería el mismo ✔
- [ ] e. Siempre imprime "Z"
- [ ] f. Siempre imprime "Q"

Pregunta 15

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuales de las siguientes sintaxis del comando for son correctas?

- [ ] a.
```
for i in `seq 1 100` do
echo "$i"
done
```
- [x] b. ✔
```
for i in `seq 1 100`; do echo "$i"; done
```
- [x] c. ✔
```
for i in $(seq 1 100); do echo "$i"; done
```
- [ ] d.
```
for $i in `seq 1 100`
do
echo "$i"
done
```
- [x] e. ✔
```
for i in `seq 1 100`
do
echo "$i"
done
```

Pregunta 16

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué imprime la siguiente secuencia de comandos? (Responder solamente el resultado de la secuencia de comandos, sin explicaciones).

```
echo 1zTrao dpaoqda | tr Taqz1rd irbuQea
```

Respuesta: Quiero aprobar ✔

Pregunta 17

Correcta

Se puntúa 1,00 sobre 1,00

¿Que información tiene la variable $#?

- (•) a. La cantidad de parametros que se enviaron ✔
- ( ) b. Ninguna es solo un comentario
- ( ) c. La cantidad de variables utilizadas en el script
- ( ) d. La cantidad de arreglos utilizados en el script

Pregunta 18

Correcta

Se puntúa 1,00 sobre 1,00

¿Con que simbolo se declara un comentario?

- a. ?
- b. //
- c. # ✔
- d. ;
- e. %

Pregunta 19

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué hace el siguiente script?

```
#!/bin/sh
a=$(wc -c xyz | cut -d' ' -f1)
expr $a / 1024 / 1024
```

- a. Comprime el archivo "xyz" a "xyz.wc" e imprime el tamaño en MiB del archivo comprimido.
- b. Falla ya que "wc" debe recibir los datos por la entrada estándar.
- c. Imprime el tamaño en MiB del archivo "xyz" ✔
- d. Cuenta la cantidad de caracteres en el string "xyz" y la pasa a MiB (obviamente da cero)
- e. Falla porque el valor calculado por "expr" no es almacenado en ninguna variable.

Pregunta 20

Correcta

Se puntúa 1,00 sobre 1,00

¿Que hace el comando "find / –name pepe 2> /dev/null" ?

- a. muestra los archivos llamados pepe, siempre y cuando tenga permisos ✔ de acceso a ellos
- b. envia la salida del comando al /dev/null
- c. crea un archivo vacio en /dev/null