Comenzado el	jueves, 3 de octubre de 2024, 20:43
Estado	Finalizado
Finalizado en	jueves, 3 de octubre de 2024, 21:10
Tiempo empleado	27 minutos 12 segundos

Pregunta **1**

Parcialmente correcta

Se puntúa como 0 sobre 1,00

Asocie los siguientes operadores de bash con sus significados o comandos equivalentes.

echo cadena | grep archivo	Busca el string "archivo" dentro del string "cadena" ✔
echo $((expresión))	Equivale a: test expresión ✖
echo cadena >> archivo	Escribe "cadena" al final de "archivo" ✔
[expresión]	Equivale a: expr expresión ✖
$(./archivo)	Equivale a: `./archivo` ✔
cadena=`./archivo`	Guarda lo que imprime "archivo" al ejecutarse adentro de la variable "cadena" ✔
echo cadena > archivo	Sobreescribe "archivo" con el string "cadena" ✔

Pregunta **2**

Correcta

Se puntúa como 0 sobre 1,00

¿Con que simbolo se declara un comentario?

a. %
b. //
c. ?
d. # ✔
e. ;

Pregunta **3**

Correcta

Se puntúa como 0 sobre 1,00

¿Cuales de las siguientes sintaxis de funciones es correcta?

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

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

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

Pregunta **4**

Correcta

Se puntúa como 0 sobre 1,00

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

- [x] 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
- [ ] e.
  for i in `seq 1 100` do
  echo "$i"
  done

Pregunta **5**

Correcta

Se puntúa como 0 sobre 1,00

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

- [x] a. if test -r /home/pepe; then echo "Tengo permisos de lectura"; fi ✔
- [x] b. ✔
  if ["$nombre" == "Maria"]
  then
  echo "Es igual"
  fi
- [ ] c.
  if ["$nombre" == "Maria"]
  then echo "Es igual"
  fi
- [ ] d. if test -d /home/pepe then echo "es un directorio" fi

Pregunta **6**

Correcta

Se puntúa como 0 sobre 1,00

¿Cuales de los siguientes usos de arreglos son correctos?

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

Pregunta **7**

Correcta

Se puntúa como 0 sobre 1,00

¿Cuales de los siguientes usos de expr es correcto?

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

Pregunta **8**

Correcta

Se puntúa como 0 sobre 1,00

¿Cuales de los siguientes usos de variables son correctos?

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

Pregunta **9**

Correcta

Se puntúa como 0 sobre 1,00

¿Que comillas se utilizan para la hacer uso de la sustitucion de comandos?

- (•) a. `` ✔
- ( ) b. ^ ^
- ( ) c. ' '
- ( ) d. " "

Pregunta **10**

Correcta

Se puntúa como 0 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. crea un archivo vacio en /dev/null
- ( ) c. envia la salida del comando al /dev/null

Pregunta **11**

Correcta

Se puntúa como 0 sobre 1,00

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

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

Pregunta **12**

Correcta

Se puntúa como 0 sobre 1,00

¿Que información tiene la variable $#?

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

Pregunta **13**

Correcta

Se puntúa como 0 sobre 1,00

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

- (•) a. Imprime los nombres de los usuarios que empiecen con la letra a ✔
- ( ) b. No imprime nada
- ( ) c. Imprime las password de los usuarios que contenga una letra a
- ( ) d. Imprime los nombre de usuario que contenga una letra a
- ( ) e. Imprime los homes de los usuarios que tienen una letra a

Pregunta **14**

Parcialmente correcta

Se puntúa como 0 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. Imprime que los valores 25 y 50 son divisibles por 25
- [x] c. Cuando el valor de i llega al 53, el script termina. ✔
- [ ] d. Imprime todos los valores del 1 al 100 divisibles por 25

Pregunta **15**

Correcta

Se puntúa como 0 sobre 1,00

¿Cualés de las siguientes son funciones correctamente definidas?

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

Pregunta **16**

Parcialmente correcta

Se puntúa como 0 sobre 1,00

¿Cuales de las siguientes equivalencias son ciertas?

a. `echo hola | cat > salida` ✔
```
# equivale a:
echo hola > salida
```

b. `find -name archivo`
```
# equivale a:
echo archivo | find -name
```

c. `cat archivo | wc -c`
```
# equivale a:
wc -c archivo | cut -d' ' -f1
```

d. `grep ac archivo` ✔
```
# equivale a:
cat archivo | grep ac
```

e. `echo hola | cat > salida`
```
# equivale a:
cat hola > salida
```

f. `find -name archivo` ✔
```
# equivale a:
find . -name archivo
```

Pregunta **17**

Incorrecta

Se puntúa como 0 sobre 1,00

Qué opciones son verdaderas respecto a la secuencia de comandos:

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

a. Si "archivo" existe y contiene el string "menta" imprime "Q", sino imprime "Z" ✖

b. Siempre imprime "Q"

c. Sin los paréntesis el resultado sería distinto ✖

d. Si "archivo" existe y contiene el string "menta" imprime "Z", sino imprime "Q"

e. Siempre imprime "Z"

f. Sin los paréntesis el resultado sería el mismo

Pregunta **18**

Correcta

Se puntúa como 0 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. Ningun resultado

c. 00:e0:7d:b4:1c:38

d. 255.255.255.255

e. fe80::2e0:7dff:feb4:1c38/64

Pregunta **19**

Correcta

Se puntúa como 0 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
for arg in $@; do
    echo $arg
done
```

c.
```
#!/bin/sh
echo $*
```

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 **20**

Parcialmente correcta

Se puntúa como 0 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
for i; do
echo $i
break
done
```

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

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

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

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

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

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

g.
```
#!/bin/sh
echo $1
```

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