EJERCICIOS SCRIPTS

Puesta en producción segura

[FECHA]

RUBEN ESQUIVEL MORON
Ciberseguridad en entornos de las tecnologías de la información

Contenido

1. Investiga en internet y encuentra alguna metodología AGILE alternativa a Scrum. Explica cómo se aplica y porqué mejora el desarrollo seguro de software.

- -Kanban: Kanban es una metodologíaía que se centra en la visualizaciónón del flujo de trabajo y la limitaciónón de la cantidad de trabajo en progreso. En lugar de dividir el trabajo en iteraciones (como en Scrum), Kanban se centra en el flujo continuo de trabajo. Los equipos utilizan un tablero Kanban para visualizar el progreso del trabajo y para identificar cuellos de botella en el proceso Kanban.

- Extreme Programming (XP): XP es una metodologíaía que se centra en mejorar la calidad del software y la capacidad de respuesta al cambio. XP introduce muchas prácticas que se han convertido en estándar en la industria de desarrollo de software, como la programación en parejas, las pruebas automatizadas y la integración continua Extreme Programming (XP).

- Feature-Driven Development (FDD): FDD es una metodología que se centra en la entrega de características completas del software. En lugar de trabajar en iteraciones, los equipos trabajan en características individuales. FDD se centra en la colaboración entre los equipos de desarrollo, los equipos de negocio y los equipos de usuarios Feature-Driven Development (FDD).

- Lean: Lean es una metodología que se centra en eliminar el desperdicio y mejorar la eficiencia. En lugar de trabajar en iteraciones, los equipos trabajan en pequeños lotes de trabajo y utilizan técnicas como el kanban y el pull system para gestionar el flujo de trabajo Lean.

2. Explica con tus palabras las diferencias existentes entre lenguajes compilados e interpretados. Pon varios ejemplos de ambos y expón en qué tipo de servicios o aplicaciones se emplean más frecuentemente.

- En los lenguajes compilados, el código fuente se traduce directamente en código de máquina, que luego se ejecuta en la máquina objetivo. Este proceso de traducción se realiza antes de la ejecución del programa, lo que significa que el código fuente no se necesita para ejecutar el programa. Algunos ejemplos de lenguajes compilados son C, C++

- En los lenguajes interpretados, el código fuente se traduce en un formato intermedio que luego es ejecutado por un programa llamado intérprete. El intérprete traduce y ejecuta el código líneo por línea mientras el programa se está ejecutando. Algunos ejemplos de lenguajes interpretados son Python, JavaScript

3. Haz una comparativa de dos o tres editores o entornos de lenguajes de programación detallando sus características principales. Explica cuál vas a utilizar en principio durante el curso y detalla porqué.

1. PyCharm:

PyCharm es un IDE muy popular desarrollado por JetBrains. Es especialmente adecuado para desarrolladores profesionales y facilita el desarrollo de grandes proyectos en Python simplilearn.com.

Características principales:

I. Soporte para bases de datos: PyCharm permite acceder a bases de datos directamente desde el IDE, lo que facilita la manipulación de datos en tu código simplilearn.com.

II. Integración con herramientas de control de versiones: PyCharm se integra con sistemas de control de versiones como Git y Mercurial, lo que facilita el seguimiento de cambios en tu código pythoncentral.io.

III. Depuración y pruebas: PyCharm tiene potentes herramientas de depuración y pruebas, lo que facilita la identificación y corrección de errores en tu código pythoncentral.io.

2. Visual Studio Code:

Visual Studio Code es un editor de código ligero y extensible desarrollado por Microsoft. Aunque no es un IDE completo, ofrece una amplia gama de extensiones que pueden convertirlo en un IDE de Python completo hackr.io.

Características principales:

I. Extensibilidad: Visual Studio Code se puede extender con una amplia gama de extensiones, lo que permite agregar nuevas funcionalidades al editor hackr.io.

II. Depuración: Visual Studio Code tiene soporte integrado para la depuración de código Python hackr.io.

III. Integración con Git: Visual Studio Code tiene una excelente integración con Git, lo que facilita el seguimiento de cambios en tu código hackr.io.

- En mi caso, utilizaré Visual Studio Code ya que tengo experiencia en dicho programa y me resulta muy como de utilizar

4. Realiza un script llamado 'holamundo.sh' que muestre por pantalla "Hola mundo".

- Crearemos un script con el comando “nano” y le pondremos de extensión “.sh” Dentro de él escribiremos #!/bin/bash para que el script pueda ser reconocido por lo que es
- Dentro del script, escribiremos “echo Hola mundo” para que lo devuelva por pantalla, y guardaremos el script
- Por último, le daremos los permisos de ejecución a nuestro script con “chmod 777 + nombre del script” y ya podremos ejecutarlo
- Como vemos en pantalla, al ejecutar el script con “./nombre:del_script.sh se ejecuta correctamente, devolviendo “Hola mundo” por pantalla

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ nano holamundo.sh
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ chmod 777 holamundo
.sh
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./holamundo.sh
hola mundo
```

5. Crear un script que muestre por pantalla los parámetros introducidos como una sola cadena.

- De nuevo, crearemos un script y le daremos sus respectivos permisos de ejecución, pero el contenido será diferente
- Escribiremos lo que se observa en la captura y le agregaremos “$*” lo cual quiere decir que se enseñará en pantalla TODOS los parámetros ofrecidos a la ejecución del script

```
 GNU nano 2.9.3                                              ejercicio5.sh

#!/bin/bash

echo "Estos son los parametros introducidos: $*"
```

- En este caso, podemos observar como al script le paso los parámetros “1,2,3,4,5,6” al ejecutarlo y observamos que los muestra por pantalla correctamente

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio5.sh 1 2 3 4 5 6
Estos son los parametros introducidos: 1 2 3 4 5 6
```

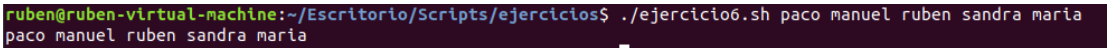
6. Crea un script el cual despliegue en pantalla todos los valores de los parámetros enviados, separados por el valor de IFS

- En este script, deberemos de dar un valor a la variable “IFS” la cual se utiliza para dictar un tipo de espaciado, ya sea un guion, barra, coma, punto…

- En este caso, lo he dejado como un espaciado sin nada, con lo que debemos especificarlo en “IFS =” ””

- Seguidamente, le decimos que enseñe los parámetros que le pasemos al script

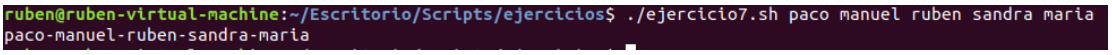
- Y al ejecutarlo, vemos como devuelve los parámetros por un espacio



7. Crea un script el cual despliegue en pantalla todos los valores de los parámetros enviados, separados por un guion.

- De nuevo, crearemos el mismo script, pero esta vez dándole un valor guion (-) a la variable IFS

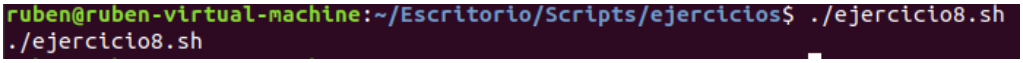
- Y aquí estaría el resultado



8. Crea un script que muestre en pantalla el nombre del mismo.

- Para crear un script que diga el nombre del mismo script, debemos simplemente mencionar un “echo “$0”” ya que eso menciona el nombre del archivo ejecutado

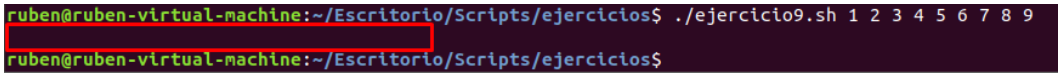
- Seguidamente lo ejecutamos y podemos ver como lo enseña por pantalla



9. ¿Qué pasaría si un Script recibe 9 parámetros y se intenta imprimir el parámetro 10?

- Para crear este script, simplemente deberemos enseñar el parámetro $10 en un “echo” lo cual en este caso, se escribe diferente, ya que contiene 2 números con lo que deberemos escribirlos entre llaves como se ve en la captura

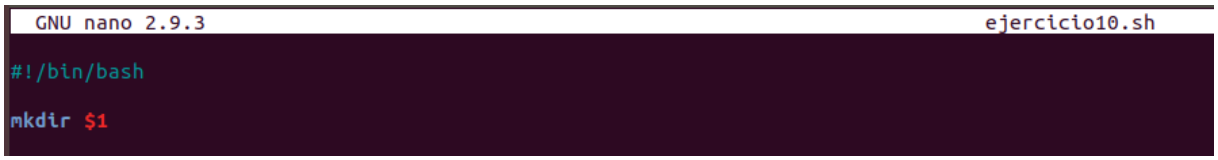
- Y si al ejecutarlo, pasandole 9 parametros, intentamos ver que pasa, observamos como no devuelve nada



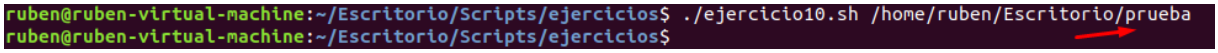
10. Escribe un script que cree un directorio a partir del primer parámetro recibido. El primer parámetro recibido debería ser, por lo tanto, la ruta correcta del directorio que queremos crear (de forma relativa o absoluta) incluyendo su nombre.

- Para crear este script, le insertaremos el comando “mkdir” el cual crea directorios en la ruta especificada a continuación

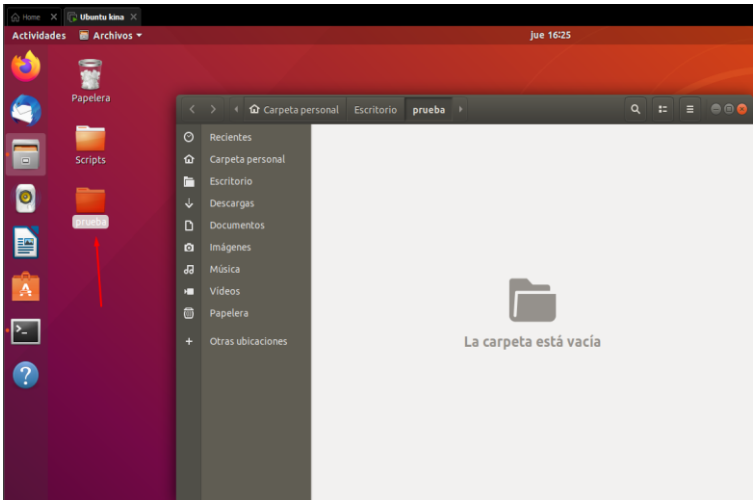
- Deberemos decir que cree la carpeta en la ruta especificada en el parámetro, con lo que seguido del “mkdir” escribiremos “$1”



- Al ejecutarlo, podremos ver como se ejecuta correctamente

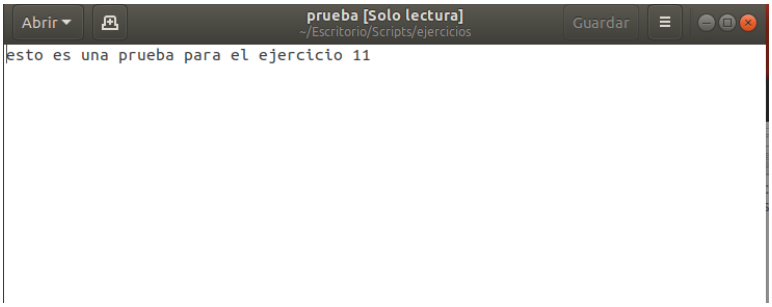


- Y podemos observar como se crea el directorio asociado al parámetro asociado al script



11. Crea un script que muestre en pantalla el contenido de un archivo a partir del segundo parámetro recibido. El segundo parámetro recibido debería ser la ruta (relativa o absoluta) a un archivo existente. Finalmente muestra el exit code del comando usado para visualizar el archivo.

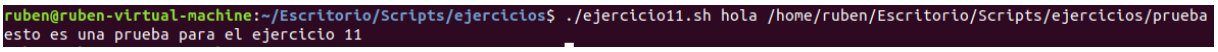
- Antes de hacer este ejercicio, deberemos crear un fichero para, posteriormente visualizarlo



- Para este script, deberemos escribir un comando “cat” ya que este muestra el contenido de un fichero

- Seguidamente, escribiremos $2 ya que nos esta pidiendo el contenido del SEGUNDO parametro

- Por ultimo al ejecutarlo, deberemos escribir dos parámetros, uno aleatorio que no sirve de nada, ya que el importante es el segundo, el cual deberemos escribir la ruta del fichero a visualizar

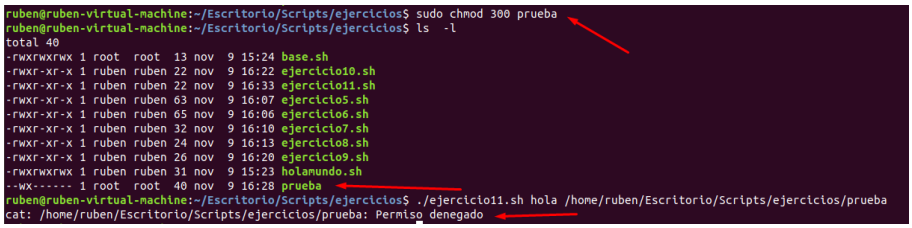


- Cuando hayas realizado el script, y comprobado que funciona correctamente, realiza la siguiente prueba: Pasa como 2º parámetro la ruta a un archivo sobre el que no tengas permiso de lectura.

- Primero deberemos quitarle los permisos de lectura al fichero con el que queramos hacer la prueba, con lo que lo haremos con el comando “chmod 300 <nombre_del_fichero>”

- Al hacerlo, nos aseguraremos con el comando “ls -l” el cual visualiza los permisos en el directorio seleccionado

- Por ultimo ejecutamos el script del ejercicio y observamos que nos da error de permisos

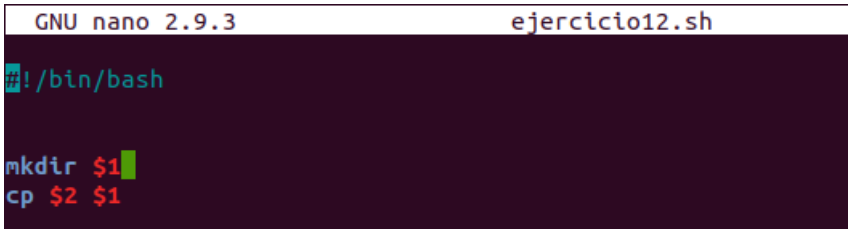


12. Crea un script que cree un directorio y después copie dentro de él un fichero. Las rutas del directorio y fichero a copiar se pasarán como parámetros. Por lo tanto, recibirá 2 parámetros:

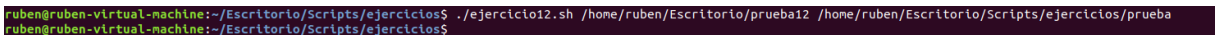
A. Primer parámetro: La ruta del directorio a crear, incluyendo su nombre (similar al ejercicio 7).

B. Segundo parámetro: La ruta de un archivo existente.

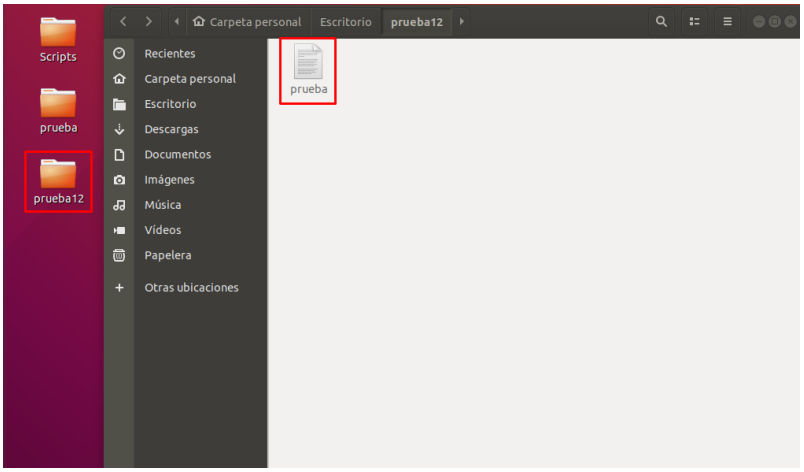
- Para crear este script, deberemos utilizar dos parámetros, el de la ruta a crear y el del archivo el cual agregaremos (el cual ya existe)
- Para ello, crearemos un directorio con “mkdir” agregándole el primer parámetro
- Seguidamente copiaremos el fichero dado por el segundo parámetro, en la ruta de la nueva carpeta creada anteriormente, para ello utilizaremos el comando “cp” el cual se utiliza de la siguiente manera “**cp <ruta_del_archivo_a_copiar> <ruta_de_destino**>”
- Con lo que al escribir el cp, deberemos asignarle la ruta del archivo existente (segundo parametro) + la carpeta creada anteriormente (primer parametro)



- Podemos observar como al ejecutarlo, no devuelve fallo



- Y al comprobarlo, la carpeta es creada y se le ha agregado el fichero mencionado



13. Crea un script que realice las siguientes tareas:

A. Si el script no ha recibido ningún parámetro, que muestre el mensaje “No has introducido ningún parámetro” y termine el script.

B. Si ha recibido algún parámetro:

- Que diga cuantos parámetros ha recibido.
- Que muestre los parámetros recibidos.

- En este script, comenzaremos utilizando el condicional “if” el cual utilizaremos para tener varias opciones, en este caso, si nuestro script NO recibe parámetros, que lo muestre y si recibe parámetros, que los muestre y los cuente

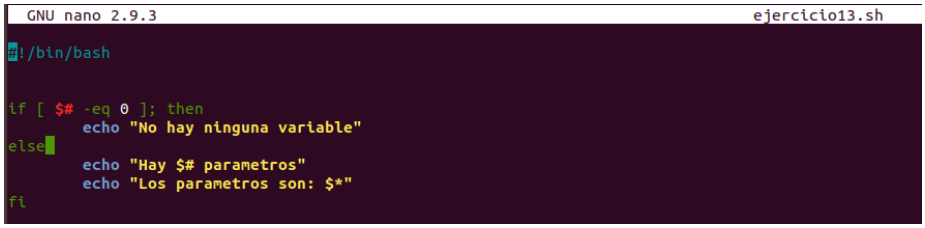
- Dentro de nuestro condicional, le estamos escribiendo $# el cual sirve para contar los parámetros asociados devolviéndolo en numero

- Seguidamente, hay escrito “-eq” el cual compara variables, estados…

- Con lo que estamos diciendo que “si la cuenta de parámetros es igual a 0…”

- Si es igual a 0, mostrará en pantalla “No hay ninguna variable”

- Si NO es igual a 0 (con lo que el script tiene asociado algún parametro) procede a hacer la segundaopcion, la cual es contar los parámetros con el “$#” y mostrarlos con el “$*” puestos en un “echo”



- Al ejecutarlo con parámetros asociados, podemos observar como muestra el conteo de los parámetros y los muestra en pantalla

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio13.sh 1 2 3 4
Hay 4 parametros
Los parametros son: 1 2 3 4
```

- Al no agregarle ningún parámetro, lo mencionará

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio13.sh
No hay ninguna variable
```

14. Modifica el script anterior, de manera que el script retorne un 1 en el caso A (después de mostrar el mensaje), y devuelva un 0 en el caso B (después de haber mostrado la información sobre los parámetros). Cuando pruebes este script ejecutándolo desde la terminal, tras ejecutarlo verifica su exit code.

- Al irnos al ejercicio anterior (explicación del ejercicio también en dicho ejercicio) a cada opción del "if" le agregaremos un "exit <numero>" para saber por que opción esta saliendo nuestro if

- En este caso, si no hay parámetro, agregaremos "exit 1" y si tiene parámetros "exit 0"

```
 GNU nano 2.9.3                                          ejercicio13.sh

#!/bin/bash


if [ $# -eq 0 ]; then
        echo "No hay ninguna variable"
        exit 1
else
        echo "Hay $# parametros"
        echo "Los parametros son: $*"
        exit 0
fi
```

- Ahora, para hacer la prueba, probaremos a ejecutarlo sin parámetros, ¿nos devolverá el resultado y seguidamente deberemos escribir "$?" el cual sirve para saber por que numero de "exit" ha ido
- También lo haremos poniendo parámetros y podremos ver que al comprobar el exit, el numero es diferente, ya que sale por uno o por otro según la salida del if

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio13.sh
No hay ninguna variable
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ $?
1: orden no encontrada
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio13.sh 1 2 3 4 5 6
Hay 6 parametros
Los parametros son: 1 2 3 4 5 6
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ $?
0: orden no encontrada
```

15. Repite el script del ejercicio 7, pero verificando que se le ha pasado un parámetro, e informando adecuadamente si el script no recibe al menos un parámetro.

- Partiendo del ejercicio 7, lo modificaremos para SI recibe parámetros, lo mencione y si NO recibe, lo mencione de manera diferente, en este caso enseñándolos por pantalla
- Como en el ejercicio 13, deberemos crear un "if" el cual vea si hay o no parámetros

```
 GNU nano 2.9.3                                                ejercicio15.sh

#!/bin/bash

IFS="-"

if [ $# -eq 0 ];
then
        echo "No has introducido ningun parametro"
else
        echo "Este Script ha recibido parametros, los cuales son: $*"
fi
```

- Lo cual al ejecutarlo, podremos ver como si le asignamos parámetros, los muestra y si no, lo dice

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio15.sh
No has introducido ningun parametro
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio15.sh 1 paco manu
Este Script ha recibido parametros, los cuales son: 1-paco-manu
```

16. Repite el ejercicio 9, pero verificando que se le han pasado al menos 2 parámetros, e informando adecuadamente si no recibe los parámetros esperados.

- Para este script, deberemos utilizar el condicional “if” pero esta vez, escribiendo varias condiciones
- Para comenzar, deberemos decir “si el numero de parámetros asociados es mayor o igual a 2 Y es menor a 9” entonces mencionar que se han pasado mínimo 2 parametros
- Para ello, utilizaremos lo siguiente

-ge → mayor o igual que…
-lt → menor que…

- Si tiene 9 parametros exactos entonces el script se ejecutará correctamente intentando mostrar el decimo parámetro el cual no existe, para ello utilizaremos:

-eq → igual que…

- Si recime menos de 2 parametros o mas de 9 entonces mostrará que no ha recibido los parámetros esperados

```
 GNU nano 2.9.3                                              ejercicio16.sh

#!/bin/bash

if [ $# -ge 2 ] && [ $# -lt 9 ]; then

echo "Se han pasado minimo 2 parametros"

elif [ $# -eq 9 ];then

echo "PROGRAMA EJECUTADO CORRECTAMENTE"
echo "${10}"

else

echo "El programa no ha recibido los parametros esperados"
fi
```

- Podemos ver como funciona correctamente

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio16.sh
El programa no ha recibido los parametros esperados
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio16.sh 1 2 3
Se han pasado minimo 2 parametros
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio16.sh 1 2 3 4 5 6 7 8 9
PROGRAMA EJECUTADO CORRECTAMENTE

ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$
```

17. Crea un script que reciba como parámetro la ruta a un archivo o directorio, y nos informe si existe. Si el script no recibe un parámetro debe informar adecuadamente.

- Para este script, usaremos de nuevo un “if” donde en la primera condición, le diremos que no ha agregado ningún parámetro como hemos hehco en ejercicios anteriores
- Dentro del “else” el cual se utiliza para la “opción contraria” en este caso, que si exista parámetro asociado, crearemos otro “if” donde comprobaremos si existe o no esa ruta aportada como parámetro
- Para ello utilizaremos:

-e →comprueba si existe o no un directorio

- Con lo que ya dependiendo si existe o no, utilizará una ruta del if u otra para mencionar si el directorio existe

```
 GNU nano 2.9.3                                              ejercicio17.sh

#!/bin/bash

if [ $# -eq 0 ];then
        echo "Nos has agregado ningun parametro"
else
        if [ -e "$1" ];then
                echo "Esta ruta de archivo/directorio existe"
        else
                echo "Esta ruta de archivo/directorio NO existe"
        fi
fi
```

- Aquí podemos ver como se ejecuta correctamente

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio17.sh /home/ruben
Esta ruta de archivo/directorio existe
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio17.sh /home/ru
Esta ruta de archivo/directorio NO existe
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio17.sh
Nos has agregado ningun parametro
```

18. Crea un script que reciba como parámetro la ruta a un archivo o directorio y, si existe, que nos indique si es un archivo o un directorio.

- Para este script, utilizaremos el anterior hecho en el ejercicio 17, el cual agregaremos otro “if” dentro de otro “if” dentro de otro “if” para comprobar si el directorio o fichero que se le ha asociado como parámetro es una cosa u otra

- En este caso utilizaremos lo siguiente:

-f → comprueba si es un fichero

- Lo agregaremos al if para comprobar si es un fichero, y si no es un fichero, es un directorio

```
 GNU nano 2.9.3                                                  ejercicio18.sh

#!/bin/bash

if [ $# -eq 0 ];then
        echo "Nos has agregado ningun parametro"
else
        if [ -e "$1" ];then
                echo "Esta ruta de archivo/directorio existe"
                        if [ -f "$1" ];then
                                echo "Esto es un archivo"
                        else
                                echo "Esto es un directorio"
                        fi
        else
                echo "Esta ruta de archivo/directorio NO existe"
        fi
fi
```

- Al ejecutarlo, podremos ver como se cumplen todas las condiciones

```
                                ruben@ruben-virtual-machine: ~/Escritorio/Scripts/ejercicios
 Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio18.sh /home
Esta ruta de archivo/directorio existe
Esto es un directorio
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio18.sh /home/ruben/Escritorio/Scripts/ejercicios/prueba
Esta ruta de archivo/directorio existe
Esto es un archivo
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio18.sh /home/ruben/Escritorio/Scripts/ejercicios/pru
Esta ruta de archivo/directorio NO existe
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio18.sh
Nos has agregado ningun parametro
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$
```

19. Modifica el script del ejercicio 12, para que verifique si existe el directorio antes de crearlo. Si existe, simplemente debe mostrar un mensaje.

- Partiendo del ejercicio 12, le agregaremos el condicional “if” para comprobar si el directorio existe

- Para ello, como hemos utilizado anteriormente, comprobaremos con “-e y -d” para ver si existe y si es un directorio

- Según su salida del if, nos dirá que el directorio existe si ya estaba creado, o nos lo creará correctamente si no lo estaba de antes

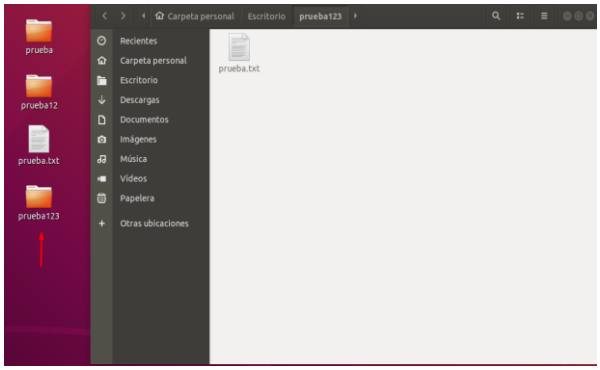
```
 GNU nano 2.9.3                                          ejercicio19.sh

#!/bin/bash

if [ -e "$1" ] && [ -d "$1" ];then
        echo "Este directorio ya existe"
else
echo "Directorio creado correctamente"
mkdir $1
cp $2 $1
fi
```

- Aquí podemos ver su ejecucion

```
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio19.sh /home/ruben/Escritorio/prueba123 /home/ruben/Escritorio/prueba.txt
Directorio creado correctamente
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$ ./ejercicio19.sh /home/ruben/Escritorio /home/ruben/Escritorio/prueba.txt
Este directorio ya existe
ruben@ruben-virtual-machine:~/Escritorio/Scripts/ejercicios$
```



20. Modifica el script anterior, para que nos informe si la creación del directorio tuvo éxito o no. Aquí tienes varias posibilidades, una de ellas es poner el comando de creación en la propia condición del if (el if evaluará el exit code del comando). Otra posibilidad es intentar la creación primero y después comprobar el valor del exit code.

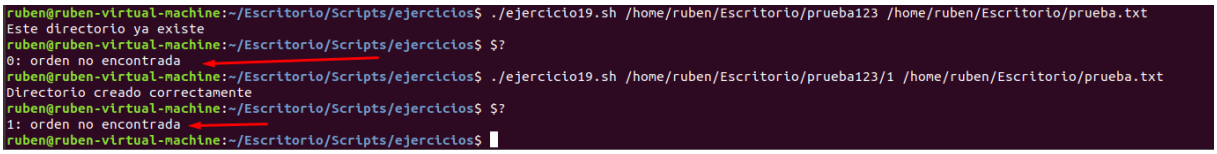
- Partiendo del ejercicio anterior, utilizaremos el comando “exit” para comprobar la salida que toma nuestro script

```
                                          ruben@ruben-virtual-machine: ~/Escritor
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
 GNU nano 2.9.3                                          ejercicio19.sh

#!/bin/bash

if [ -e "$1" ] && [ -d "$1" ];then
        echo "Este directorio ya existe"
        exit 0
else
echo "Directorio creado correctamente"
mkdir $1
cp $2 $1
exit 1
fi
```

- Comprobación de casos



OPCIONAL: Haz un repositorio en la nube de GitHub y clona tu repositorio/carpeta local con los ejercicios de scripts alli. Aporta la dirección del repositorio (tiene que ser público para ser visto por el profesor).

No olvides insertar comentarios en el código de los scripts explicando lo que creas oportuno para que la solución del problema sea más legible.

Link a GitHub

FORMATO DE ENTREGA: **T1A1_1erApellido2ºApellido_nombre**