1.- Un archivo de requisitos, **requirements.txt**, suele contener una lista de paquetes (Linux) o módulos (Python) que se pretenden instalar de forma automática.

Crea un script que lea el contenido de **/media/paquetesLinux/requirements.txt** y vaya instalando, uno a uno, todos los paquetes (con la opción –y para que no tengamos que intervenir).

Es posible que a la hora de instalar alguno no se instale porque no sea un paquete conocido.

El script debe ir “anotando” en un fichero (/media/incidenciasLinux) todo lo que va sucediendo, es decir, el paquete xxxx se ha instalado bien o el paquete yyy no es un paquete o no se ha podido instalar

**#!/bin/bash**

**FICHERO=”/media/paquetesLinux”**

**INCIDENCIAS=”/media/incidenciasLinux”**

**if [ ! -f $FICHERO ]; then**

**echo “$FICHERO no existe”**

**exit 1**

**fi**

**for PAQUETE in $(cat $FICHERO) ; do**

**sudo apt-get install $PAQUETE**

**if [ $? -eq 0 ] ; then**

**echo “El paquete $PAQUETE se ha instalado bien” >> “$INCIDENCIAS”**

**else**

**echo “El paquete $PAQUETE no es un paquete o no se ha podido instalar” > “$INCIDENCIAS”**

**fi**

**done**

2.-Crea un script que solicite el nombre de un directorio por teclado. Si existe, debe mostrar **el nombre de los ficheros** que contenga indicando al final el número de los mismos. Si no existe, debe mostrar un mensaje y terminar.

Eres root y este script lo puede ejecutar todo el mundo, pero no modificarlo.

**#!/bin/bash**

**read -p “Introduzca el nombre de un directorio: ” DIRECTORIO**

**CONTADOR\_FICHEROS=0**

**if [ -d “$DIRECTORIO” ]; then**

**for FICHEROS in “$DIR”/\*; do**

**if [ -f “$FICHEROS” ]; then**

**echo “$(basename “$FICHEROS”)”**

**((CONTADOR\_FICHEROS++))**

**fi**

**done**

**echo “Tenemos $CONTADOR\_FICHEROS ficheros”**

**else**

**echo “$DIRECTORIO no es un directorio”**

**fi**

3.- Contesta a estas cuestiones:

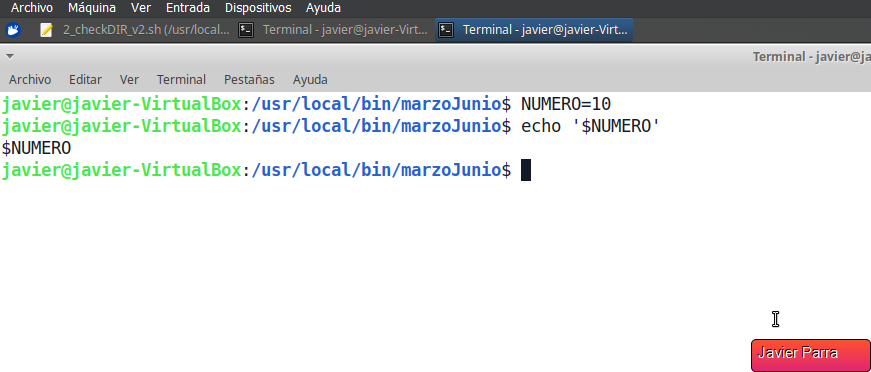
A.- Explica lo que hace el siguiente comando:

**#mkdir directorio && cd directorio && touch f**

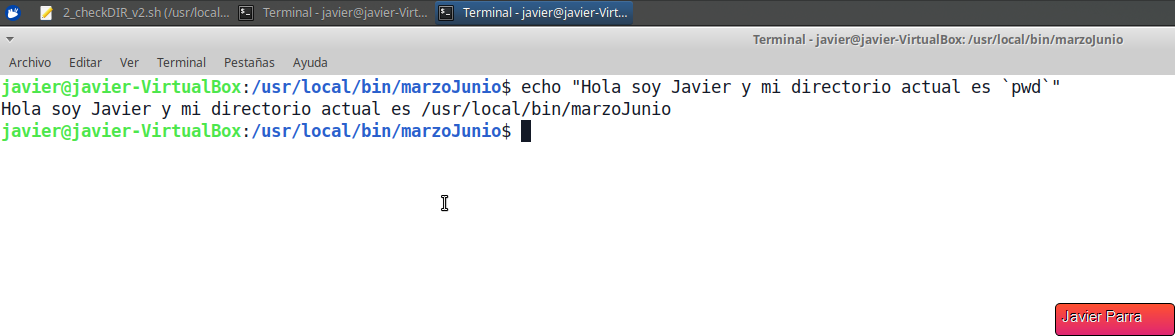
**El siguiente comando crea el directorio llamado directorio y luego con && significa “y” ejecuta el comando cd para cambiarse al directorio directorio y luego por último otro “y” y crea dentro del directorio f. Si el primer comando falla, los demás no se van a ejecutar.**

1. - Diferencia entre las comillas simples ‘ ’ y las comillas invertidas ` ` en Linux. Ejemplos.

**Las comillas simples ‘ ’ sacan el texto literal por pantalla. Por ejemplo si declaramos una variable que contiene un nombre, si después con un echo y comillas simples de la variable, no se mostrara lo que contiene si no, el texto literal.**



**Las comillas invertidas ` ` ejecutan un comando o interpretan un comando dentro de una cadena de texto.**



1. - Principales estructuras de control **iterativas** en bash. Ejemplos.

**Bucle While y For**

**For LINEA in $(cat $FICHERO) ; do**

**Echo $LINEA**

**Done**

**$FICHERO=”ejemplo.txt”**

**While read -r LINEA ; do**

**Echo “Leyendo la línea $LINEA”**

**Done < $FICHERO**

4.- Observa este script

**#!/bin/bash**

**numero1=10**

**numero2=0**

**resultado=$((numero1 / numero2))**

**echo "El resultado de la división es: $resultado"**

Introduce, **dentro del código,** lo necesario para que se vea, paso a paso, la ejecución del mismo y se vaya mostrando en pantalla la salida.

¿Qué se irá viendo en pantalla?

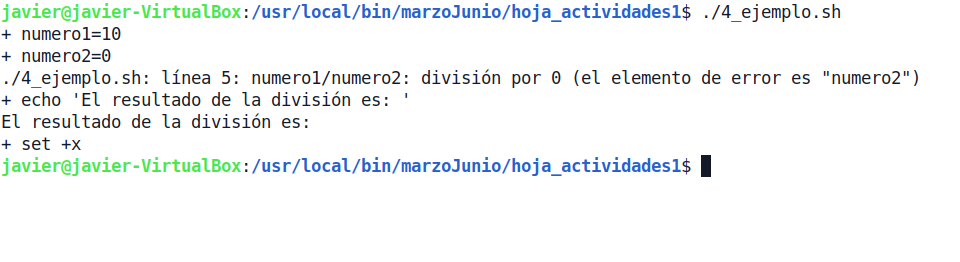
**Iremos viendo la ejecución del script línea por línea.**

¿Habrá algún tipo de error?

**Si, un error de lógica, por que en la división no se puede dividir entre 0.**

¿Qué tipos de errores conoces?

**Errores de tiempos de ejecución ,errores en la sintaxis, errores en variables no definidas y errores en la lógica.**



5.- Nos han pedido que realicemos un script básico para que instale, configure para que se inicie al arrancar el sistema e inicie en ese momento, el paquete que le viene por la línea de comandos.

**Observación:** un paquete con la opción –y se instala sin “hacer preguntas”.

**Observación:** puedes ayudarte, entre otros, del comando

**systemctl list-unit-files** que muestra todos los servicios disponibles

Ejemplo de ejecución: **#adm\_pte samba**

**#!/bin/bash**

**# Se commprueba los argumentos que vienen por la línea de comandos**

**if [ $# -eq 0 ] ; then**

**echo "Se necesita proporcionar un paquete"**

**exit 1**

**fi**

**PAQUETE=$1**

**# Comprobamos si existe el servicio instalado en el sistema**

**if systemctl list-unit-files | grep "$PAQUETE" ; then**

**echo "El paquete $PAQUETE esta instalado"**

**else**

**read -p "El paquete $PAQUETE no está instalado, ¿Deseas instalarlo? (S/N): " RESPUESTA**

**if [ $RESPUESTA = "S" ] ; then**

**echo "Instalando $PAQUETE"**

**sudo apt-get install -y $PAQUETE**

**sudo systemctl enable $PAQUETE**

**fi**

**fi**

6.- Realizar un script que solicite por teclado el nombre de un servicio y muestre al usuario un menú (se facilita) con las opciones que puede hacer con dicho servicio.

Si el servicio no existe, se muestra un mensaje y se termina.

Ejemplo de menú

**echo "Seleccione la acción a realizar:"**

**echo "1. Iniciar servicio"**

**echo "2. Detener servicio"**

**echo "3. Reiniciar servicio"**

**read opcion**

**#!/bin/bash**

**pausa()**

**{**

**read -p "Pulsa ENTER para continuar..." TECLA**

**}**

**OPCION="0"**

**while [ "$OPCION" != "5" ]; do**

**clear**

**echo "Seleccione una acción a realizar:"**

**echo "1.Iniciar servicio"**

**echo "2.Detener servicio"**

**echo "3.Reiniciar servicio"**

**echo "4.Salir del programa"**

**read -p "Su opcion..." OPCION**

**case ${OPCION} in**

**1)**

**read -p "Introduce el nombre del servicio que quieres iniciar: " INICIAR\_SERV**

**sudo systemctl start $INICIAR\_SERV**

**if [ $? -eq 0 ]; then**

**echo "El servicio $INICIAR\_SERV se ha iniciado correctamente"**

**else**

**echo "Se ha producido un error al iniciar el servicio $INICIAR\_SERV"**

**fi**

**pausa**

**;;**

**2)**

**read -p "Introduce el nombre del servicio que quieres detener: " DETENER\_SERV**

**sudo systemctl stop $DETENER\_SERV**

**if [ $? -eq 0 ]; then**

**echo "El servicio $DETENER\_SERV se ha detenido correctamente"**

**else**

**echo "Se ha producido un error al detener el servicio $DETENER\_SERV"**

**fi**

**pausa**

**;;**

**3)**

**read -p "Introduce el nombre del servicio que quieres reiniciar: " REINICIAR\_SERV**

**sudo systemctl restart $REINICIAR\_SERV**

**if [ $? -eq 0 ]; then**

**echo "El servicio $REINICIAR\_SERV se ha reiniciado correctamente"**

**else**

**echo "Se ha producido un error al reiniciar $REINICIAR\_SERV"**

**fi**

**pausa**

**;;**

**4)**

**echo "Has salido del programa, Hasta la vista..."**

**;;**

**\*)**

**echo "No se ha encontrado tu opción, Por favor, escoje una de la lista"**

**pausa**

**;;**

**esac**

**done**

7.- Simular el comando **grep** de Linux en su forma más básica. Nuestro script debe recibir 2 argumentos: la palabra a buscar y el fichero en el que busca.

No se hace nada si no se reciben 2 argumentos o si no existe el fichero.

Nuestro script, cuando se ejecute, debe ir mostrando la línea que contenga la palabra introducida.

Al final, debe mostrar el número de líneas que ha mostrado.

**#!/bin/bash**

**# Verificar si se proporcionaron dos argumentos**

**if [ $# -ne 2 ]; then**

**echo "Debes de introducir dos argumentos, una palabra y un fichero"**

**exit 1**

**fi**

**# Verificar si el archivo existe**

**if [ ! -f "$2" ]; then**

**echo "El archivo $2 no existe."**

**exit 2**

**fi**

**# Contador para el número de líneas que contienen la palabra**

**contador\_lineas=0**

**# Contador para el número de palabras que contienen la linea**

**contador\_palabras=0**

**# Iterar sobre cada línea del archivo y buscar la palabra**

**while read -r linea; do**

**# Buscar la palabra en la línea**

**if [[ $linea == \*"$1"\* ]]; then**

**((contador\_lineas++))**

**# Array cuyo resultado es la expansion de la variable linea**

**palabras=($linea)**

**# Iterar sobre cada palabra en la línea**

**for palabra in "${palabras[@]}"; do**

**# Comparar cada palabra con la palabra buscada**

**if [[ $palabra == \*"$1"\* ]]; then**

**((contador\_palabras++))**

**fi**

**done**

**echo "$linea"**

**fi**

**done < "$2"**

**echo "Número de líneas que contienen "$1": $contador\_lineas"**

**echo "Número total de palabras que coinciden con "$1": $contador\_palabras"**

8.- En determinadas ocasiones necesitamos vaciar el contenido de un fichero, pero sin eliminarlo. El comando “*truncate --size=0 file*” vacía de contenido el fichero indicado (file en este caso). Crea el script **vaciar\_fichero** que reciba por la línea de comandos el nombre del fichero cuyo contenido queremos vaciar.

Antes de eliminarlo, preguntaría al usuario si realmente quiere hacerlo ya que es una acción irreversible.

En principio todo el mundo lo podría ejecutar, pero realmente sólo root lo puede hacer. No hacemos nada si no recibimos un fichero o este no existe.

**#!/bin/bash**

**# Se comprueban el numero de argumentos**

**if [ $# -ne 1 ] ; then**

**echo "Se necesita proporcionar el nombre de un fichero a vaciar"**

**exit 1**

**fi**

**# Variable que contiene el fichero a vaciar, como primer argumento**

**FICHERO=$1**

**# Preguntamos al usuario si de verdad quiere vaciar el fichero**

**read -p "¿Estas seguro de querer vaciar el contenido del fichero $FICHERO?, esta acción es irreversible, ¿Desea continuar? (s/N)" RESPUESTA**

**if [ "$RESPUESTA" = "S" ] ; then**

**echo "Vaciando el contenido del fichero $FICHERO...."**

**truncate --size=0 file $FICHERO**

**if [ $? -eq 0 ] ; then**

**echo "Se ha vaciado correctamente el fichero $FICHERO"**

**else**

**echo "¡Ups!, Ha habido un problema al vaciar el fichero $FICHERO"**

**fi**

**fi**

9.- Crea un script, ***del\_img.sh***, que automatice la eliminación de ficheros del directorio **/logos** cuyo tamaño **supere los 100 KiB** y **no hayan sido modificados en los últimos 15 días.**

Este script sólo lo puede ejecutar **root**. En caso contrario, se debe almacenar, sin eliminar nada, en **/logos/logs/del\_imagenes.info**, el identificador (UID) del usuario que lo ha ejecutado, así como la fecha y hora en la que se ha producido siguiendo este formato:

*El usuario cuyo UID es “UID” ha intentado eliminar, a las hh:mm:ss del día dd:mm:aaaa ficheros de /logos*.

El script, si es ejecutado como root, debe asegurarse en principio de que existe /logos. Si no es así, debe anotar en del\_imagenes.info esta incidencia.

Este script se debe ejecutar todos los días a las 21:30.

Explica todo el proceso de creación del script, dónde lo harías, planificación, etc.

**#!/bin/bash**

**# Directorio sobre donde va a trabajar el script**

**DIRECTORIO="/logos"**

**# Fichero de información con avisos de eliminación**

**FICHERO\_INFO="/logos/logs/del\_imagenes.info"**

**# Variable para guardar la fecha y hora actual**

**FECHA\_ACTUAL=$(date +%H:%M:%S-%d/%m/%Y)**

**# Verificar si el usuario es root**

**# sudo permite ejecutar el comando tee con privilegios sudo si no se ejecuta el script con privilegios de root**

**# tee me permite escribir la salida del echo al fichero y con el argumento -a "append" no me lo sobreescribe, es decir va agregando las lineas al fichero**

**# llevo la salida del comando tee que básicamente es la misma que muestro con echo al sumidero, para que directamente la escriba en el fichero del\_imagenes.info**

**if [ $UID -ne 0 ]; then**

**echo "El usuario cuyo UID es $UID ha intentado eliminar ficheros de /logos. Fecha y hora de la incidencia: $FECHA\_ACTUAL" | sudo tee -a "$FICHERO\_INFO" > /dev/null**

**echo "Este script debe ser ejecutado como root."**

**exit 1**

**fi**

**# Verificar si el directorio existe**

**if [ ! -d "$DIRECTORIO" ]; then**

**echo "El directorio $DIRECTORIO no existe." >> "$FICHERO\_INFO"**

**exit 1**

**fi**

**echo "Este script borrará del directorio $DIRECTORIO los ficheros superiores a 100KiB y con un tiempo de modificación superior a 15 días..."**

**# Le decimos al usuario si desea proceder con la eliminación**

**read -p "¿Desea continuar? (s/N): " RESPUESTA**

**# Si el usuario teclea la tecla S o s procedemos a borrar los ficheros con los criterios pedidos**

**if [ "$RESPUESTA" = "S" ] || [ "$RESPUESTA" = "s" ]; then**

**echo "Eliminando los archivos superiores a 100KiB y con 15 días sin modificar..."**

**find "$DIRECTORIO" -type f -size +100k -mtime +15 -exec rm {} \;**

**if [ $? -eq 0 ]; then**

**echo "Se han borrado los archivos correctamente $FECHA\_ACTUAL" >> "$FICHERO\_INFO"**

**else**

**echo "Ocurrió un error al eliminar los archivos"**

**fi**

**else**

**echo "Eliminación cancelada por el usuario"**

**fi**