1.- El uso de disco duro por parte de los usuarios hay que controlarlo. El comando ***sudo du –s directorio*** devuelve el tamaño en bytes del directorio dado. Por ejemplo:



Realizar un script, **size\_homes**, que

a) muestre en pantalla el **nombre de los directorios** de conexión de los usuarios existentes en el sistema que superen el umbral establecido en 10000 bytes, así como el **número total** de directorios que lo superan.

b) muestre en pantalla el nombre y tamaño del directorio que ocupa más disco.

**#!/bin/bash**

**# Establecemos el umbral en 10000 bytes**

**UMBRAL=10000**

**# Creamos las variables a 0 para contar el numero de directorios que superan el tamaño maximo**

**TOTAL\_SUPERAN=0**

**MAX\_SIZE=0**

**MAX\_DIR=""**

**# Con el bucle for recorremos todos los subdirectorios del directorio /home**

**for DIRECTORIO in /home/\* ; do**

**# Comprobamos si es un directorio**

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

**# Obtener el tamaño del directorio**

**SIZE=$(du -s "$DIRECTORIO" | cut -f1)**

**# Mostramos los directorios que superen el umbral**

**if [ "$SIZE" -gt "$UMBRAL" ]; then**

**echo "Directorio que supera el umbral: $DIRECTORIO"**

**((TOTAL\_SUPERAN++))**

**fi**

**# Actualizamos el directorio con el tamaño máximo**

**if [ "$SIZE" -gt "$MAX\_SIZE" ]; then**

**MAX\_SIZE="$SIZE"**

**MAX\_DIR="$DIRECTORIO"**

**fi**

**fi**

**done**

**# Mostramos el número total de directorios que superan el umbral**

**echo "Total de directorios que superan el umbral: $TOTAL\_SUPERAN"**

2.- Queremos generar un informe de actividad del servidor sobre el uso de…

* **Porcentaje Disco Usado (raíz)**
* **Memoria utilizada**
* **Número de procesos**

Los comandos que nos dan esta información son:

**Disco: El comando df –h | grep ‘/$’ devuelve:**

**/dev/sda1 33G 9,0G 22G 29%** /

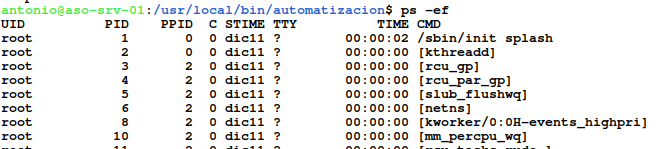
Habría que guardar, en este caso, el **29%**

**Memoria**: El comando **free –m** devuelve lo siguiente



Habría que guardar, en este caso, 580 (en MiB).

**Número de procesos**: ps –ef devuelve todos los procesos en ejecución. Habría que guardar el número de los mostrados sin contar el encabezado.



Nuestro script debe crear un fichero, en **/infomes**, con el nombre informe\_actividad + dia\_en\_el\_que\_se\_hace\_el\_informe, por ejemplo, *informe\_actividad\_12\_12\_2023.txt*, con la información obtenida cuando se ejecute el script (no hay que borrar nada del fichero informe\_actividad\_....). Contenido**:**

**Informe de Actividad 12/12/2023**

**--------------------------------------------**

**Uso de Memoria: 580 MB**

**Uso de Disco: 29%**

**Número de Procesos: 219**

**#!/bin/bash**

**# Este script genera un informe de los recursos utilizados del servidor**

**# Variable con el directorio donde guardar el fichero**

**DIRECTORIO="/informes"**

**# Variable con la fecha actual**

**FECHA\_ACTUAL=$(date +%d\_%m\_%Y)**

**# Obtenemos el uso de espacio en disco**

**ESPACIO\_DISCO=$(df -h | grep '/$'| tr -s " " | cut -d" " -f3)**

**# Obtenemos el uso de memoria ram**

**USO\_RAM=$(free -m | tr -s " " | cut -d" " -f3 | sed -n '2p')**

**# Obtenemos los procesos en ejecución**

**PROCESOS\_EJECUCION=$(ps -ef | tr -s " " | cut -d" " -f2 | wc -l)**

**# Creamos el fichero en el direcotio /informes con la información**

**sudo touch "$DIRECTORIO/informe\_actividad\_$FECHA\_ACTUAL"**

**echo "Informe de Actividad $FECHA\_ACTUAL" >> "$DIRECTORIO/informe\_actividad\_$FECHA\_ACTUAL"**

**echo "----------------------------------" >> "$DIRECTORIO/informe\_actividad\_$FECHA\_ACTUAL"**

**echo "Uso de Memoria: $USO\_RAM" >> "$DIRECTORIO/informe\_actividad\_$FECHA\_ACTUAL"**

**echo "Uso de Disco: $ESPACIO\_DISCO" >> "$DIRECTORIO/informe\_actividad\_$FECHA\_ACTUAL"**

**echo "Número de Procesos: $PROCESOS\_EJECUCION" >> "$DIRECTORIO/informe\_actividad\_$FECHA\_ACTUAL"**

1. - Crea un script, *delFileDir.sh*, que elimine, pidiendo confirmación, todos los ficheros de cada uno de los directorios que recibe por la línea de comandos (puede recibir uno o varios argumentos). Si no recibe nada, aviso y se sale.

**#!/bin/bash**

**# Este script borra todos los ficheros de los directorios que se soliciten por la línea de comandos**

**# Control de argumentos**

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

**echo "No se han proporcionado argumentos. Uso: $0 <directorio>"**

**exit 1**

**fi**

**# Bucle for para recorrer la linea de comandos**

**for DIRECTORIO in "$@"; do**

**# Comprobamos si existen los directorios**

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

**echo "El directorio $DIRECTORIO no existe"**

**else**

**read -p "¿Desea eliminar los ficheros de los direcorios $DIRECTORIO? (s/N): " RESPUESTA**

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

**echo "Eliminando los ficheros del direcorio $DIRECTORIO..."**

**find "$DIRECTORIO" -type f -exec rm -r {} \;**

**else**

**echo "Operación cancelada por el usuario, no se ha eliminado nada"**

**fi**

**# Comprobamos si se ha eliminado correctamente los ficheros de los directorios solicitados**

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

**echo "Se han eliminado correctamente los ficheros de $DIRECTORIO"**

**else**

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

**fi**

**fi**

**done**

1. - Realiza un script que muestre el nombre de la distribución de Linux que tenemos instalada, así como el núcleo instalado. Sólo debemos ver esta información.

**#!/bin/bash**

**# Este script te muestra la versión de tu kernel de Linux y el nombre de la distro que estas utilizando**

**# Versión del kernel de linux**

**KERNEL\_VERSION=$(uname -r)**

**# Distribución de Linux**

**DISTRO\_NAME=$(lsb\_release -d | cut -f2)**

**# Mostramos la version del kernel de linux junto a el nombre de la distribución**

**echo "Estás usando la versión $KERNEL\_VERSION del kernel de Linux y la distribución $DISTRO\_NAME de GNU/Linux"**

5.- Quiero simular el comando **head** de Linux. Mi script, **head\_iniciales** (head\_ajld en mi caso), debe recibir 2 o más argumentos en la línea de comandos: el primero es un número entero que indica el número de líneas a mostrar (mayor que 1 y menor a 24), el 2º argumento y siguientes (en caso de haberlos) es el nombre de los ficheros que queremos procesar. Habrá que comprobar que ese o esos ficheros existen. Ejemplo de ejecución:



**#!/bin/bash**

**# Este script muestra el numero de líneas de un fichero**

**# Comprobamos que nos llegan dos argumentos por la línea de comandos**

**# Función para mostrar las líneas de los ficheros**

**mostrar\_primeras\_lineas() {**

**ARCHIVO=$2**

**NUM\_LINEAS=$1**

**CONTADOR=0**

**# mostrar lineas**

**while read LINEA; do**

**echo "$LINEA"**

**((CONTADOR++))**

**# salir después de mostrar las primeras N Líneas**

**if [ "$CONTADOR" -eq "$NUM\_LINEAS" ]; then**

**break**

**fi**

**done < "$ARCHIVO"**

**}**

**# Obtenemos el número de líneas a mostrar**

**NUM\_LINEAS=$1**

**# Comprobamos el número de argumentos por la línea de comandos**

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

**echo "Uso: $0 <nºde lineas> <fichero1> <fichero2>"**

**exit 1**

**fi**

**# Verificamos que el primer argumento que es el números de líneas a mostrar es un entero**

**if [[ "$NUM\_LINEAS" =~ ^[0-9]+$ ]]; then**

**# Verificamos que el número de líneas es válido (mayor que 1 y menor a 24)**

**if [ "$NUM\_LINEAS" -lt 1 ] || [ "$NUM\_LINEAS" -gt 24 ]; then**

**echo "Se debe proporcionar un número entre 1 y 24"**

**exit 2**

**fi**

**fi**

**# Con un bucle for iteramos sobre los ficheros y mostramos las primeras líneas**

**# Con ${@:2} le decimos al bucle for que vamos a recorrer toda la línea de comandos pero que vamos a empezar a partir del segundo argumento con :2**

**for ARCHIVO in "${@:2}"; do**

**if [ -f "$ARCHIVO" ]; then**

**echo "--> $ARCHIVO (primeras $NUM\_LINEAS líneas) <--"**

**mostrar\_primeras\_lineas $NUM\_LINEAS $ARCHIVO**

**echo ""**

**else**

**echo "Error: EL archivo $ARCHIVO no existe."**

**echo ""**

**fi**

**done**

**6.- Añade, a nuestro fichero de funciones, estas funcionalidades**

1.- Queremos diseñar una función**, validar\_IP(),** que valide el formato de una dirección IP que es solicitada por teclado.

2.- En nuestra organización, admitimos nombres de usuarios que contienen letras mayúsculas, minúsculas, guiones bajos, guiones normales y dígitos.

Diseñar una función**, validar\_usuario()** para validar nombres de usuarios. Hay que tener en cuenta que no pueden empezar por dígito y la longitud del mismo oscila entre 7 y 12 caracteres.

3.- Queremos diseñar una función**, validar\_formato\_fecha**, que valide el formato de una fecha que es *dd-mm-aaaa*, es decir, 2 dígitos para el día, dos para el mes y cuatro para el año separados por un guion.

Esta función muestra un mensaje en pantalla tanto si la fecha tiene un formato correcto como si no. Además, cuando termine devuelve un 0 si es un formato correcto y un 1 si no es válido

Ejemplo de llamada a la función

**validar\_formato\_fecha "13-0-2023"**🡪 mal (1)

**validar\_formato\_fecha "13-05-2023"** 🡪 bien (0)

4.- Queremos diseñar una función**, validar\_formato\_correo()** para validar direcciones de correo electrónico: en la 1ª parte del correo admite caracteres alfanuméricos, puntos, guiones bajos, porcentajes y los caracteres + y -.

Después de la @, los mismos que en la 1ª parte excepto el + y el -.

Detrás del punto, al menos 2 caracteres alfabéticos.

Esta función muestra un mensaje en pantalla tanto si el email tiene un formato correcto como si no. Además, cuando termine devuelve un 0 si es un formato correcto y un 1 si no es válido.

Ejemplo de llamada

read -p "Introduce una dirección de correo electrónico: " EMAIL

**verificar\_correo\_electronico "$EMAIL"**