**Presentamos a continuación unos ejemplos de automatización de tareas típicos en cualquier sistema Linux**

**1.- Revisión de Logs de Autenticación**

Crea un script que revise logs de autenticación y detecte intentos de inicio de sesión fallidos.

**Ayuda**

Podemos buscar en **/var/log** el fichero que esté relacionado con la autenticación en el sistema.

**Observación**: inicia sesión con una cuenta de usuario (o con varios) e introduce mal la contraseña unas cuantas veces.

**#!/bin/bash**

**FICHERO="/var/log/auth.log"**

**while read -r LINEA ; do**

**MENSAJE=$(cat "$FICHERO" | grep failure | tr -s " " | cut -d" " -f7,8 | cut -d";" -f1 | cut -d" " -f2 | sed -n '2p')**

**if [[ "$LINEA" == "$MENSAJE" ]]; then**

**echo "Fallo de Inicio de Sesión. Línea: $LINEA"**

**else**

**echo "No se encontró ningún inicio de sesión incorrecto"**

**fi**

**done < "$FICHERO"**

**2.- Limpieza de Archivos Temporales:**

Estudia el siguiente comando

***$find /ruta/a/temporales -type f -mtime +7 -exec rm {} \;***

¿Qué hace?

Vamos a crear un script y para probarlo, crearemos un directorio de prueba con ficheros más antiguos y más actuales.

**#!/bin/bash**

**# Obtenemos una fecha de modificación de un fichero. Tiene que ser una de 7 dias de antiguiedad**

**# FECHA\_MODIFICACION=$(stat -c %y <nombre\_fichero\_antiguo>)**

**# Aplicar fecha de modificacion al fichero**

**# sudo touch -d "$FECHA\_MODIFICACION" <nombre\_fichero\_al\_que\_aplicar>**

**DIRECTORIO="home/ricardo/pruebas"**

**for FICHERO in $(ls "$DIRECTORIO"/\*); do**

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

**read -p "Se borrarán los ficheros temporales con 7 días de antiguedad, ¿Desea continuar? (s/n): " R**

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

**echo "Borrando ficheros con antiguedad de 7 días..."**

**find "$DIRECTORIO" -type f -mtime +7 -exec rm {} \;**

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

**echo "Se han borrado correctamente los ficheros en $DIRECTORIO"**

**else**

**echo "Hubo un problema al borrar los ficheros"**

**fi**

**else**

**echo "Operación cancelada"**

**fi**

**else**

**echo "$DIRECTORIO no existe o no es un directorio"**

**fi**

**done**

**3.- Script de Respaldo y Compresión**

Estudia el siguiente script y explica lo que realizaría

***#!/bin/bash***

***fecha=$(date +"%d-%m-%Y")***

***tar -czf backup\_$fecha.tar.gz /ruta/a/copiar1 /ruta/a/copiar2***

**El script empieza guardando en la variable fecha, la fecha actual con la expansión del comando date con una salida formateada.**

**Desupés compacta con el comando tar y comprime con gzip el contenido de los directorios que se desea respaldar. El nombre del backup será backup\_ y la expansión de la variable fecha que contiene la fecha actual.**

Ahora, programa este script para que respalde /home todos los domingos del año a las 14:55. Las copias en /copias

***tar -czf /copias/home\_$fecha.tar.gz /home/***

**$ crontab -e**

**55:14 7 \* \* /usr/local/bin/backup.sh**

**!/bin/bash**

**fecha=$(date + "%d-%m-%Y")**

**tar -czf /copias/home\_$fecha.tar.gz /home/**

**4.- Monitoreo del Uso del Espacio en Disco**

Queremos realizar un script que, cuando se ejecute, monitorice el espacio que ocupa un determinado directorio (será la partición en la que se creó) y si supera un determinado umbral, emitir una alerta.

Ayuda: **df directorio**



**#!/bin/bash**

**ROJO="\033[0;31m"**

**VERDE="\033[0;32m"**

**RESET="\033[0m"**

**DIRECTORIO="/usr/local/bin"**

**ESPACIO=$(df "$DIRECTORIO" | tr -s " " | cut -d" " -f3 | sed -n '2p')**

**UMBRAL="4237154"**

**if [ $ESPACIO -gt $UMBRAL ]; then**

**echo -e "${ROJO}Atención: $DIRECTORIO supera el umbral establecido en $UMBRAL${RESET}"**

**else**

**echo -e "${VERDE}$DIRECTORIO no supera el umbral.${RESET}"**

**fi**

**5.- Escaneo de Puertos abiertoe en un Rango de IPs**

Utiliza nmap para realizar un escaneo de determinados puertos en un rango de direcciones IP dado.

Por ejemplo, los puertos del 1 al 100 en los equipos 192.168.50.[1-50]

**#!/bin/bash**

**nmap -p 1-100 192.168.1.1-50**

**6.- Estado de los procesos.**

El comado ps -aux muestra mucha información sobre los estados de los procesos. La columna “**STAT”** en la salida del comando “ps” proporciona información sobre el estado actual de un proceso. Aquí están algunos de los valores más comunes que puedes encontrar:

- R (Running): El proceso está en ejecución o listo para ejecutarse.

- S (Sleeping): El proceso está en reposo. Esto indica que está esperando a que ocurra un evento, como una interrupción de tiempo de espera.

- D (Uninterruptible Sleep): El proceso está en un estado de sueño ininterrumpible, generalmente esperando algún evento relacionado con la entrada/salida, como una lectura de disco.

- Z (Zombie): El proceso ha terminado, pero aún aparece en la tabla de procesos. Este estado generalmente indica un problema de gestión de procesos por parte del proceso padre.

- T (Stopped): El proceso ha sido detenido (suspendido) por una señal, como cuando se ejecuta `Ctrl+Z` en la terminal.

Si tenemos una Z quiere decir que tenemos un proceso “zombie”.

Crear un script que detecte, en caso de existir, procesos zombis.





**#!/bin/bash**

**# Busca los procesos zombie y muestra el PID y el nombre del proceso.**

**zombies=$(ps -eo pid,state,comm | grep " Z")**

**if [[ -n "$zombies" ]]; then**

**echo "Procesos zombies detectados:"**

**echo "$zombies" | while read -r line; do**

**PID=$(echo "$line" | tr -s ' ' | cut -d ' ' -f 1)**

**COMANDO=$(echo "$line" | tr -s ' ' | cut -d ' ' -f 3)**

**echo "PID: $PID, Proceso: $COMANDO"**

**done**

**else**

**echo "No se han detectado procesos zombies"**

**fi**

**Presentamos a continuación una serie de scripts relacionados con la automatización de tareas relacionadas con usuarios, grupos, diectorios de trabajo, etc.**

1.- Mejorar el script **addgrupos.sh** de tal forma que, cuando vaya procesando cada línea:

a.- Compruebe, antes de crear el grupo si este existe ya.

b.- No añada grupos cuyo nombre sea mayor a 15 caracteres.

c.- Indique si no se especifica el grupo (línea en blanco)

**Fichero de prueba: grupos.txt (en rojo las filas que se consideran inválidas)**

*alumnado*

*profesorado*

*direccion*

*pas*

*alumnado*

*secretaria*

*abcdefghijklmnopqrst*

*jefatura*

**Salida**

*Procesando la fila: alumnado*

*Grupo alumnado añadido al sistema*

*….*

*Procesando la fila:*

*Fila anulada. No se ha especificado el grupo*

*….*

*Procesando la fila: alumnado*

*El grupo ya existe. No lo creamos*

*….*

*Procesando la fila: abcdefghijklmnopqrst*

*El grupo abcdefghijklmnopqrst supera los 15 caracteres y por tanto no se crea.*

**#!/bin/bash**

**FICHERO="grupos.txt"**

**for LINEA in $(cat "$FICHERO"); do**

**echo "Procesando la fila: $LINEA"**

**if [ -z "$LINEA" ]; then**

**echo "La linea está vacía"**

**elif [[ ! "$LINEA" =~ ^[A-Za-z]{1,15}$ ]]; then**

**echo "El grupo $LINEA supera los 15 caracteres, por tanto no se va a crear"**

**elif cat "/etc/group" | grep $LINEA; then**

**echo "El grupo $GRUPO ya existe"**

**else**

**echo "Creando el grupo $LINEA"**

**sudo groupadd "$LINEA"**

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

**echo "El grupo $LINEA se ha creado correctamente"**

**else**

**echo "Error al crear el grupo $LINEA"**

**fi**

**fi**

**Done**

2.- Mejorar el script **addusuarios.sh**, de tal forma que compruebe si:

a.- Existe el grupo.

b.- Existe el usuario.

c.- La longitud de la cuenta de usuario es superior a 15 caracteres en cuyo caso no se añadirá.

d.- Falta algún campo, en cuyo caso no se hará nada.

Al igual que en el ejercicio anterior, ir mostrando en pantalla la fila procesada.

**Fichero de prueba: usuarios.csv (en rojo las filas que se consideran inválidas)**

*angela, alumnado,123*

*david, alumnado, 456*

*juan,blandengues,123*

*mario, profesorado, 789*

*david,profesorado,123*

*paraquevaleestatonteria,alumnado,123*

*pedro,,123*

*,algo,*

**#!/bin/bash**

**FICHERO="usuarios.csv"**

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

**echo "$FICHERO no existe"**

**exit 1**

**fi**

**while IFS="," read -r LINEA ; do**

**USUARIO=$(echo $LINEA | cut -d"," -f1)**

**GRUPO=$(echo $LINEA | cut -d"," -f2)**

**CLAVE=$(echo $LINEA | cut -d"," -f3)**

**if [ -z "$USUARIO" ] || [ -z "$GRUPO" ] || [ -z "$CLAVE" ]; then**

**echo "Error, falta algún campo en la línea"**

**elif [[ ! "$USUARIO" =~ ^[A-Za-z]{1,15}$ ]]; then**

**echo "Error, la longitud de la cuenta de usuario es mayor a 15 caracteres"**

**elif cat "/etc/passwd" | grep "$USUARIO" ; then**

**echo "El usuario ya existe en el sistema"**

**elif cat "/etc/group" | grep "$GRUPO" ; then**

**echo "El grupo ya existe en el sistema"**

**else**

**echo "Creando el usuario $USUARIO..."**

**# Creamos el usuario y le añadimos al grupo que pertenece y la shell**

**/usr/sbin/useradd -m -g $GRUPO -s /bin/bash $USUARIO**

**# Al usuario le ponemos su contraseña**

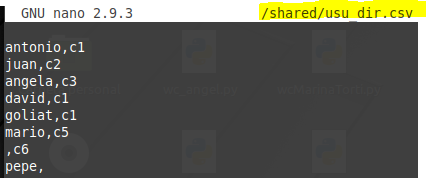
**echo "$USUARIO:$CLAVE" | chpasswd**

**fi**

**done < "$FICHERO"**

3.- Automatiza la creación de determinadas carpetas a determinados usuarios en su directorio home.

Estas carpetas vendrán en un fichero de texto **/shared/usu\_dir.csv** cuyo primer campo es el usuario y el segundo, el nombre de la carpeta.

Ejemplo de contenido de /shared/usu\_dir.csv. 

Significa que en /home/antonio hay que crear el directorio c1, en /home/juan, el directorio c2 y así sucesivamente.

Hay que tener en cuenta que:

* **Solo lo puede ejecutar root**
* **No se hace nada si en cada línea de las procesadas falta algún campo.**
* **No se crea nada si el usuario no existe,**

**#!/bin/bash**

**FICHERO="/shared/usu\_dir.csv"**

**# Solo root puede ejecutar el script**

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

**echo "Permiso denegado. Necesitas ejecutarlo con sudo"**

**exit 1**

**fi**

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

**echo "$FICHERO no existe"**

**exit 2**

**fi**

**for LINEA in $(cat "$FICHERO") ; do**

**USUARIO=$(echo "$LINEA" | cut -d"," -f1)**

**CARPETA=$(echo "$LINEA" | cut -d"," -f2)**

**if [ -z "$USUARIO" ] || [ -z "$CARPETA" ]; then**

**echo "Error, falta algún campo en el fichero $FICHERO"**

**elif ! cat "/etc/passwd" | grep "$USUARIO" ; then**

**echo "El usuario $USUARIO no existe en el sistema"**

**else**

**echo "Creando la carpeta $CARPETA para el usuario $USUARIO..."**

**mkdir /home/$USUARIO/$CARPETA**

**chown -R $USUARIO:$USUARIO /home/$USUARIO/$CARPETA**

**chmod 775 -R /home/$USUARIO/$CARPETA**

**fi**

**done**

4.- Nos han solicitado que realicemos un script, **suspicious.sh,** que compruebe si el contenido de determinados ficheros contiene alguna de las siguientes palabras que consideramos sospechosas. Estas son: **hacker**, **cracker**, **phreaker** y **carder**.

Los ficheros que hay que analizar se encuentran en **/analizar**. Todos aquellos que se consideren sospechosos deben moverse al directorio **/kofiles** y los que no se consideren sospechosos, se moverán a **/okfiles**.

**#!/bin/bash**

**DIR\_OK="/okfiles"**

**DIR\_KO="/kofiles"**

**DIR\_ANALIZAR="/analizar"**

**# Declaramos un array que contine las palabras clave sospechosas**

**PALABRAS\_BAN=("hacker" "cracker" "phreaker" "carder")**

**for FICHERO in "$DIR\_ANALIZAR"/\* ; do**

**# Declaramos una variable con el valor de sospechosos a 0**

**SOSPECHOSO=0**

**# Con el bucle while recorremos las lineas del fichero encontrado**

**while read -r LINEA ; do**

**for PALABRA in "${PALABRAS\_BAN[@]}"; do**

**if [[ "$LINEA" == \*"$PALABRA"\* ]]; then**

**# Si la línea del fichero contiene alguna de las palabras sospechosas del array, cambiamos el valor de la variable sospechoso a 1**

**SOSPECHOSO=1**

**# No hay mas condiciones que evaluar, por lo tanto se sale de este bucle for**

**# Ya no hay más líneas que cumplan la condición del if, por lo tanto se sale del bucle for**

**break 2**

**fi**

**done**

**done < "$FICHERO"**

**if [ $SOSPECHOSO -eq 1 ]; then**

**echo "$FICHERO es sospechoso"**

**sudo mv "$FICHERO" "$DIR\_KO"**

**else**

**echo "$FICHERO no es sospechoso"**

**sudo mv "$FICHERO" "$DIR\_OK"**

**fi**

**done**

5.- (Parecido al ejercicio 3)

Codifica un script que cuando se ejecute procese el fichero input.txt y haga lo siguiente:

1.- Debe crear la carpeta **/Proy-Pru**. Si esta existiera, la debe borrar junto con su contenido y después crearla.

2.- Cree, en la carpeta **/Proy-Pru** las carpetas asociadas a los **proyectos** que aparezcan en las líneas que sean válidas. Si ya se ha creado una carpeta, mostrar el correspondiente mensaje y no crearla.

**Formato de input.txt: emple,rol,proyecto**

Ejemplo de contenido

**emp2,Operario,ProyectoB**

**emp5,Operario,ProyectoB**

**emp4,Operario,ProyectoB**

**emp1,Operario,ProyectoA**

**emp1,Asistente,ProyectoA**

**emp4,Ingeniero,ProyectoA**

**emp7,Ingeniero,ProyectoB**

**emp6,Operario,ProyectoA**

**emp2,Operario,ProyectoA**

**emp7,Ingeniero,ProyectoA**

**emp7,Ingeniero,ProyectoA**

**emp9,Director,ProyectoA**

**emp1,Director,ProyectoC**

**emp10,Ingeniero**

**emp11**

**Observaciones**

1.- **Roles válidos**: Operario, Asistente e Ingeniero

2.- No se consideran válidas aquellas líneas en las que aparezca un rol inexistente o les falte algún campo.

3.- Cada vez que se procese una línea, se debe mostrar el mensaje que la describa, así como las acciones que se van realizando. Por ejemplo:

***Procesando la fila: Usuario=emp2, Rol=Operario, Proyecto: ProyectoB***

***Creando la carpeta ProyectoB***

***Procesando la fila: Usuario=emp5, Rol=Operario, Proyecto: ProyectoB***

***La carpeta ProyectoB ya ha sido creada.***

***….***

***Procesando la fila: Usuario=emp9, Rol=Director, Proyecto: ProyectoA***

***El rol Director no existe. Fila anulada***

***Procesando la fila: Usuario=emp1, Rol=Director, Proyecto: ProyectoC***

***El rol Director no existe. Fila anulada***

***Procesando la fila: Usuario=emp10, Rol=Ingeniero, Proyecto:***

***No se ha especificado proyecto. Fila anulada***

***Procesando la fila: Usuario=emp11, Rol=, Proyecto:***

***No se han especificado ni rol ni proyecto.***

**#!/bin/bash**

**# 1.- Debe crear la carpeta /Proy-Pru. Si existe la borra y después la crea.**

**CARPETA="/Proy-Pru"**

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

**sudo rm -r "$CARPETA"**

**fi**

**sudo mkdir "$CARPETA"**

**# 2.- Cree en la carpeta /Proy-Pru las carpetas asociadas a los proyectos en las líneas válidas.**

**FICHERO="input.txt"**

**ROLES\_VALIDOS=("Operario" "Asistente" "Ingeniero")**

**# Leer el fichero línea por línea**

**while IFS=, read -r EMPLEADO ROL CARPETA\_PROYECTO; do**

**echo "Procesando la fila: Usuario=$EMPLEADO, Rol=$ROL, Proyecto=$CARPETA\_PROYECTO"**

**if [ -z "$CARPETA\_PROYECTO" ]; then**

**echo "No se ha especificado un proyecto. Fila anulada"**

**continue**

**fi**

**if ! [[ " ${ROLES\_VALIDOS[@]} " =~ " $ROL " ]]; then**

**echo "El rol $ROL no existe. Fila anulada"**

**continue**

**fi**

**if [ -z "$EMPLEADO" ]; then**

**echo "No se ha especificado un empleado. Fila anulada"**

**continue**

**fi**

**if [ -z "$ROL" ]; then**

**echo "No se ha especificado un rol. Fila anulada"**

**continue**

**fi**

**# Verificar si la carpeta ya existe**

**if [ -d "$CARPETA/$CARPETA\_PROYECTO" ]; then**

**echo "La carpeta $CARPETA\_PROYECTO ya ha sido creada"**

**else**

**sudo mkdir "$CARPETA/$CARPETA\_PROYECTO"**

**echo "Creando la carpeta $CARPETA\_PROYECTO"**

**fi**

**done < "$FICHERO"**