Laboratorio de administración y gestión de redes y sistemas
Examen práctico. 11 de enero de 2024.
Grado en Ingeniería Telemática, Universidad Rey Juan Carlos

Ejercicio 1 (5 puntos)

Escribe un script en Python 3 que borre todos los ficheros binarios ejecutables que encuentre en un directorio y sus subdirectorios, recursivamente. Esto se podría resolver de muchas formas (p.e. bibliotecas *libmagic* y *pathlib* de Pyhton o el comando *find* de Linux combinado con el comando *rm*), pero queremos hacerlo con las técnicas vistas en la asignatura: llamar desde python a comandos de la shell de Linux. Escribe el script `~/borra_exe.py` según lo detallado a continuación:

1. El script recibirá un argumento: el directorio desde el que partirá. P.e el punto (directorio actual), el *home* o cualquier otro.

2. El script invocará el comando `find` pasando el argumento recibido. Ejemplo: si el usuario ejecuta `~/borra_exe.py /tmp`

   el script invocará `find /tmp`

3. Como probablemente sabes, *find* recorrerá de forma recursiva todos los ficheros a partir del punto indicado y devolverá una lista de líneas, una por cada fichero, cada una con el trayecto relativo de un fichero.

4. El script usará el comando *file* para saber si cada uno de esos fichero es un binario ejecutable o no. Si el informe de file contiene el texto *LSB executable*, lo consideraremos un binario ejecutable, y si no, no. (Observa que ignoramos los permisos del fichero)

5. Observa que puede dar la casualidad de que un programa se llame `LSB executable`. En este caso tu script también debe funcionar correctamente.

6. Para cada fichero, el script escribirá una línea similar a estas, según corresponda.

   ```
   borrando ./practica03/multiplos
   conservando ./practica03/multiplos.pas
   ```

7. El programa también indicará cuantos ficheros ha borrado y cuántos ha conservado.

8. En caso de que el script haya sido lanzado con la opción `-n` o con la opción `--dry-run`, lo que ejecutará será un *ensayo* o *ejecución de prueba*, esto es, mostrará prácticamente la misma salida que una ejecución normal, pero no llegará a borrar nada. En caso de ejecución de prueba, el script escribirá en una primera línea un mensaje de advertencia, el que veas oportuno, explicando esta circunstancia.

9. Es necesario que uses la librería *optparse*.

10. Prueba tu script con los ficheros que encontrarás en `~/ejemplos.tgz`

Ejercicio 2 (5 puntos)

En este ejercicio prepararás un contenedor que codifique en ISO 8859-1 (latin1) todos los ficheros con extensión .txt que estén en el directorio `/tmp` de una máquina. Tal y como se detalla a continuación:

1. El contenedor estará basado en una imagen llamada `ex24`.
2. El contenedor estará construido a partir de ubuntu 22.04, con los paquetes actualizados a la última versión disponible. Tendrá instalado el paquete *recode*. Y ningún otro paquete ni ninguna otra configuración distinta a la de un contenedor docker por omisión. Si añades paquetes o configuración adicional, tendrás una penalización en la nota.
3. El nombre del contenedor será el nombre de la imagen, añadiendo como prefijo las primeras 4 letras de tu nombre de usuario en el laboratorio, y añadiendo el sufijo *01*. Ejemplo: Si tu nombre de usuario fuera *jperez*, el contenedor se llamaría `jperex2401`.
4. El contenedor se ejecutará en una máquina VirtualBox lanzada desde Vagrant, cuyo *project Directory* será el directorio `~/vbox` de la máquina del laboratorio.
5. Esta máquina Vagrant deberá provisionarse con el fichero de configuración

   `~/vbox/VagrantFile`, que tedrás que preparar. Será muy parecido al que usaste durante el curso, pero

   - Ahora la máquina se llama vbox, no vbox01
   - Estará actualizada, tendrá instalado docker.io, pero no tendrá ningún usuario adicional (solo el usuario vagrant que trae por omisión)
6. El contenedor tendrá un montaje de tipo *bind*, el directorio `/tmp` de la máquina *vbox* estará montado en el directorio `/var/tmp` del contenedor.
7. Al ejecutarse el contenedor, la herramienta recode convertirá desde utf8 hasta latin1 los ficheros `/var/tmp/*.txt` del contenedor (o lo que es lo mismo, los ficheros `tmp/*.txt` de la máquina vbox). Si estos ficheros no se pueden leer, no hace falta que hagas nada especial: aparecerá el mensaje de error que genere *recode*.
8. En clase no hemos visto el uso de comodines con *recode*, pero no tienes que hace nada especial: en vez de pasar como argumento el nombre de un fichero, puedes escribir p.e. `*.txt`
9. Observa que el contenedor no hará nada de manera interactiva (te puede venir bien preparar una versión interactiva provisional para probarlo todo, pero la versión que entregues, no debe lanzar ninguna sesión interactiva)
10. Los ficheros necesarios para todo esto seguirán un convenio muy parecido pero no idéntico al que hemos venido usando durante toda la asignatura. (Reemplazando *jper* por las primeras 4 letras de tu login)

    ```
    ~/vbox/VagrantFile
    ~/vbox/ex24/construye.sh
    ~/vbox/ex24/lanza_jperex2401.sh
    ~/vbox/ex24/context/Dockerfile
    ~/vbox/ex24/context/entrypoint.sh
    ```

11. Esto es: ahora la carpeta de la máquina Vagrant se llama *vbox*. No *vbox01*. Y cuelga directamente del *home* de la cuenta examen. No de la carpeta *lagrs*.
12. Enséñaselo al profesor cuando acabes: prepara dos o tres ficheros de texto utf-8 de prueba vbox, recodifícalos con tu contenedor, demuestra que han pasado a tener codificación latin1.