Laboratorio de administración y gestión de redes y sistemas
Examen práctico
20 de enero de 2023.

Grado en Ingeniería Telemática, Universidad Rey Juan Carlos

Ejercicio 1 (5 puntos)

En este ejercicio prepararás un contenedor Docker según la siguiente especificación:

1. Deseamos que al ponerse en marcha el contenedor, el usuario *root* tenga una sesión de shell abierta.

2. El usuario *root* tendrá disponible en el alias *c* el comando *clear*.

3. Como sabes, esto podríamos conseguirlo editando el fichero `.bashrc`. Pero no queremos definir el alias aquí, sino en un fichero separado.

4. El fichero `.bashrc` del usuario root dentro de un contenedor con ubuntu 22.04 ya está preparado para esto, es decir, ya tiene previsto que se puedan definir los alias en un fichero separado. El propio fichero `.bashrc` explica cómo conseguir esto. Entra de forma interactiva en el contedor, busca en `.bashrc` estas instrucciones y síguelas.

5. Para poder leer el `.bashrc`, instala *sobre la marcha*, de manera interactiva, tu editor de texto preferido dentro del contenedor. No en la imagen. Explica en un comentario en el Dockerfile cómo has hecho esto.

6. El contenedor estará construido a partir de ubuntu 22.04, con los paquetes actualizados a la última versión disponible. No tendrá ningún otro paquete ni ninguna configuración adicional (si añades algo *de sobra*, tendras una ligera penalización)

7. La imagen del contenedor se llamará `alias`. Los contenedores basados en esta imagen (en este caso solo uno), tendrán como prefijo parte de tu nombre de usuario, y como sufijo, un número. Si tu nombre de usuario fuera *jperez*, el prefijo sería *jper*. El sufijo, *01*. Por tanto el contenedor se llamaría `jperalias01`. Sustituye *jper* por los primeros 4 caracteres de tu nombre de usuario.

8. Siguiendo un convenio como el que hemos venido usando durante toda la asignatura, los nombres de los ficheros serán:

```
~/lagrs.enero.23/alias/construye.sh
~/lagrs.enero.23/alias/lanza_jperalias01.sh
~/lagrs.enero.23/alias/context/Dockerfile
~/lagrs.enero.23/alias/context/entrypoint.sh
~/lagrs.enero.23/alias/context/EL_OTRO_FICHERO
```

   Reemplazando:

   - *jper* por las primeras 4 letras de tu login)
   - `EL_OTRO_FICHERO` por el nombre del fichero que contenga el alias, siguiendo las instrucciones indicadas en el `.bashrc`

9. Ten en cuenta que cuando dentro del contenedor es necesario un fichero oculto (por tanto, empezando por punto), dentro del directorio contexto es conveniente que el fichero no esté oculto (no empiece por punto).

   Para ello basta con escribir en Dockerfile algo como esto:

   ```
   COPY  fichero_tal /bla/bla/bla/.fichero_tal
   ```

Cuando acabes, enseña el resultado al profesor.

Solución

Leyendo el fichero `.bashrc` encontramos lo siguiente:

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

#if [ -f ~/.bash_aliases ]; then
#    . ~/.bash_aliases
#fi
```

Esto es, que podemos escribir los alias en un fichero separado, de nombre `~/.bash_aliases`. El sistema se encargará de ejecutarlo con las tres líneas de shell similares a las que aparecen a continuación (exactamente esas no, porque aparecen comentadas). Son las mismas instrucciones que usamos para executar el `.bashrc` en el `.bash_profile`: si existe el fichero `~/.bash_aliases`, que se ejecute con `source`. Por tanto, los ficheros que necesitamos son los siguientes:

Fichero `construye.sh`

```
#!/bin/bash
docker build -t alias context
```

Fichero `lanza_jperalias01.sh`

```
#!/bin/bash
docker run -it --rm --name jperalias01 alias
```

Fichero `Dockerfile`

```
FROM ubuntu:22.04
COPY bash_aliases /root/.bash_aliases
COPY entrypoint.sh /
ENTRYPOINT ["/entrypoint.sh"]

#Para instalar un editor, pe. vim, ejecutamos dentro de la sesión del contenedor
#apt update;
#apt upgrade -y
#apt install -y vim
```

Fichero `entrypoint.sh`

```
#!/bin/bash
/bin/bash
```

Fichero `bash_aliases`

```
alias c='clear'
```

Ejercicio 2 (5 puntos)

Escribe un programa en python llamado `~/lagrs.enero.23/enlaces.TULOGIN.py` que resuelva lo indicado a continuación.

1. Deseamos un pequeño informe sobre los enlaces simbólicos que haya en cierto directorio.
2. El directorio sobre el que se emitirá el informe será el primer argumento recibido por el script. Si el script no recibe ningún argumento, trabajará sobre el directorio actual. Ejemplo, si el usuario ejecuta

   `./enlaces.py /tmp/aa`

   se mostrará un informe sobre el directorio `/tmp/aa`
3. El informe estará basado en la ejecución del comando `ls -l`. Supongamos que el comando genera una salida como esta

   ```
   -rw-rw-r-- 1 koji koji  0 ene  9 14:08 a
   -rw-rw-r-- 1 koji koji  0 ene  9 14:08 b
   lrwxrwxrwx 1 koji koji  1 ene  9 14:08 c -> b
   lrwxrwxrwx 1 koji koji 10 ene  9 13:52 hosts -> /etc/hosts
   ```

   En este caso, el informe sería similar a este texto:

   ```
   En el directorio /tmp/aa hay 2 enlaces simbólicos
   El fichero c que apunta a b
   El fichero hosts que apunta a /etc/hosts
   ```

4. Si no hubiera ningún enlace, el mensaje sería similar a

   ```
   En el directorio /tmp/aa no hay enlaces simbólicos
   ```

Observaciones

1. El programa usará la librería *optparse* para capturar el nombre del directorio (o para detectar que el usuario no lo ha pasado y por tanto se tomará el actual)

Cuando acabes, enseña el resultado al profesor.

Solución

```python
#!/usr/bin/env python3
import sys
import subprocess
import os
from optparse import OptionParser

def ejecuta_comando(comando):
    resultado = 0
    comando_troceado=comando.split() #elemento 0 -> ls, elemento 1 -> -l
    try:
        salida=subprocess.check_output(comando_troceado)
    except subprocess.CalledProcessError:
        sys.stderr.write("La orden ha producido un error\n")
        raise SystemExit

    salida=salida.decode("utf-8")
    lineas=salida.split("\n")
    lineas.pop(0)  # Quitamos la primera línea, "total nnn"
    return lineas


def filtra_enlaces(lineas):
    """
    Recibe una lista de lineas (de ls -l)
    Devuelve solo las que contengan enlaces simbólicos, esto es,
    las que comiencen por "l".

    Si no recordamos esto, podríamos haber identificado las
    líneas que contengan la subcadena "->" en el campo 8 (un fichero
    podría tener este nombre, pero aparecería 'escapado')
    """
    rval = []

    for linea in lineas:
        if len(linea) != 0 and linea[0] == "l":
            rval.append(linea)
    return rval


def procesa_linea(linea):
    """
    Recibe una línea de un listado conteniendo un enlace.
    Devuelve el texto correspondiente del informe.
    """

    campos=linea.split()
    fichero_original = campos[8]
    fichero_enlace = campos[10]
```

```
    msg = "El fichero {} que apunta a {}".format(
        fichero_original, fichero_enlace)
    return msg


def revisa_directorio(directorio):
    if not os.path.isdir(directorio):
        msg = "No existe el directorio {}".format(directorio)
        raise Exception(msg)
    return


def main():
    uso = "Uso: %prog [directorio]"
    parser = OptionParser(uso)
    opciones, argumentos = parser.parse_args()
    if len(argumentos) > 1:
        parser.error("Numero incorrecto de argumentos")

    if len(argumentos) == 0:
        directorio = "."
        comando = "ls -l "

    if len(argumentos) == 1:
        directorio = argumentos[0]
        revisa_directorio(directorio)

    comando = "ls -l " + directorio

    lineas = ejecuta_comando(comando)
    lineas = filtra_enlaces(lineas)

    msg = "En el directorio {} hay {} enlaces simbólicos".format(
        directorio, len(lineas))
    print(msg)
    for l in lineas:
        print(procesa_linea(l))
    return


if __name__ == "__main__":
    main()
```