Programación y Administración de Sistemas
Práctica 2. Expresiones regulares para programación de la shell.

Convocatoria de junio (curso 2023/2024)

Víctor Manuel Vargas Yun / Mª Isabel Jiménez Velasco

18 de marzo de 2024

Resumen

Esta serie de ejercicios se os entregan para que podáis practicar y profundicéis vuestros conocimientos de *bash* de cara al examen de prácticas. Estos ejercicios no se entregan, la evaluación de la práctica 2 se realizará mediante ejercicios similares a los expuestos en este guion. Para evitar problemas al ejecutar tus ejercicios de cara al examen, asegúrate de que todos los scripts que realices funcionen correctamente en los ordenadores de la UCO o conectándote por `ssh` al `ts.uco.es`. Para cualquier duda de los ejercicios, por favor, escribid en el foro del moodle o enviad un correo a la dirección vvargas@uco.es o isajimenez@uco.es

1. `ejercicio1.sh`

Crear un *script* que ejecute los comandos adecuados de *grep* que permitan realizar las siguientes tareas sobre el fichero de ejemplo *peliculas.txt*. El *script* recibirá el nombre del fichero por la línea de comandos. Hacer las comprobaciones de argumentos necesarias.

1. Mostrar el título de las peliculas que tengan una longitud de 4 palabras. (Los títulos siempre comienzan por "> ")
2. Mostrar las duraciones de películas que sean supeiores a 1h 45min. La duración de las películas se muestra con uno o más dígitos al principio de la línea, un espacio y la secuencia "min".
3. Contar cuántas películas hay de cada país, suponiendo que el país siempre está encerrado entre guiones (por ejemplo, "-Estados Unidos-").
4. Mostrar aquellas palabras que contengan la letra "d","l" o "t", una vocal y la misma letra (por ejemplo, "Universidad", "expectativas", "dedicarse",etc).
5. Mostrar todas aquellas líneas que terminen con 3 puntos ("...") y no empiecen por espacio, utilizando el operador de repetición {}.

A continuación, se muestra un ejemplo de la salida de este *script* :

```
i72jivem@VTS3:~/PAS/p2$ ./ejercicio1.sh
Argumentos erroneos . Uso: ./ejercicio1.sh <fichero_peliculas>.

i72jivem@VTS3:~/PAS/p2$ ./ejercicio1.sh  peli.txt
Se esperaba un fichero del tipo peliculas.txt

i72jivem@VTS3:~/PAS/p2$  ./ejercicio1.sh peliculas.txt
1) Titulo de las peliculas que tengan una longitud de 4 palabras:
> Kong: La Isla Calavera
> Es por tu bien
```

```
11
12 2) Duraciones superiores a 1h 45min
13 1hr 59min
14 2hr 17min
15 2hr 09min
16 1hr 48min
17
18 3) Numero de peliculas por pais
19       2 -Espana-
20       3 -Estados Unidos-
21       1 -Rusia-
22
23 4) Lineas que contengan d, l o t , una vocal , y misma letra
24 soldados
25 ciudad
26 realidad
27 expectativas
28 escondidos
29 lila
30 oportunidad
31
32 5) Lineas que acaben con 3 puntos y no empiecen por espacios
33 Reparto: Tom Hiddleston, Samuel L. Jackson, Brie Larson...
34 Reparto: Hugh Jackman, Patrick Stewart, Dafne Keen...
35 Reparto: Jos  Coronado, Javier C mara , Roberto  lamo ...
36 Reparto: Marta Etura, Elvira Minguez, Nene...
37 Reparto: Matthew McConaughey, Reese Witherspoon, Seth MacFarlane...
38 Reparto: Documentary, Gabe Polsky...
```

2. `ejercicio2.sh`

Utilizando *sed*, hacer un script que, dado el fichero de texto *películas.txt* (recibido por línea de comandos), elimine las líneas vacías, los subrayados y lo formatee de la siguiente manera por cada película:

```
Título: XXX
| -> Fecha de estreno: XXX
| -> Director: XXX
| -> Reparto: XXX
| -> Duración: XXX
```

Además, el script deberá mostrar la salida por terminal y además guardarla en un *.txt* denominado *peliculas_formateadas.txt*, utilizando el comando *tee*.

A continuación, se muestra un ejemplo de la salida de este *script* :

```
1  i72jivem@VTS3:~/PAS/p2$./ejercicio2.sh
2  Argumentos erroneos . Uso: ./ejercicio2.sh <fichero_peliculas>.
3
4  i72jivem@VTS3:~/PAS/p2$ ./ejercicio2.sh pelis.txt
5  Se esperaba un fichero del tipo peliculas.txt
6
7  i72jivem@VTS3:~/PAS/p2$ ./ejercicio2.sh peliculas.txt
8  Titulo:  Kong: La Isla Calavera
9  |->Fecha de estreno: (10/03/2017)
10 |->Director: Jordan Vogt-Roberts
11 |->Reparto: Tom Hiddleston, Samuel L. Jackson, Brie Larson...
12 |->Duracion: 1hr 59min
13 Titulo:  Logan
14 |->Fecha de estreno: (03/03/2017)
15 |->Director: James Mangold
16 |->Reparto: Hugh Jackman, Patrick Stewart, Dafne Keen...
17 |->Duracion: 2hr 17min
18 Titulo:  Es por tu bien
19 |->Fecha de estreno: (24/02/2017)
20 |->Director: Carlos Theron
21 |->Reparto: Jose Coronado, Javier Camara, Roberto Alamo...
22 |->Duracion: 1hr 33min
23 Titulo:  El guardi n Invisible
24 |->Fecha de estreno: (03/03/2017)
25 |->Director: Fernando Gonzalez Molina
26 |->Reparto: Marta Etura, Elvira Minguez, Nene...
```

```
|->Duracion: 2hr 09min
Titulo:   Canta !
|->Fecha de estreno: (22/12/2016)
|->Director: Garth Jennings
|->Reparto: Matthew McConaughey, Reese Witherspoon, Seth MacFarlane...
|->Duracion: 1hr 48min
Titulo:  Red Army
|->Fecha de estreno: (29/01/2015)
|->Director: Gabe Polsky
|->Reparto: Documentary, Gabe Polsky...
|->Duracion: 1hr 25min

Fichero peliculas_formateadas.txt creado con exito.
```

3. `ejercicio3.sh` traceroute -w 1 (IP)

La eficiencia y rapidez con la que los datos viajan de un punto a otro es crucial para mantener una experiencia de usuario fluida y efectiva. Cada vez que accedemos a un sitio web o utilizamos un servicio en línea, nuestros datos inician un viaje a través de múltiples nodos en la red, conocidos como "saltos", antes de alcanzar su destino final. Este proceso, aunque ocurre en milisegundos, es fundamental para todo, desde cargar una página web hasta transmitir un video en tiempo real. Para asegurar una navegación óptima y eficiente, es esencial que estos saltos sean lo más directos y rápidos posible. Utilizando **traceroute**, una herramienta de diagnóstico que traza la ruta que los paquetes de datos toman hasta un destino especificado, podemos obtener una visión detallada de cada salto en el camino.

Desarrolla un *script* de Bash que ejecute traceroute para una lista de direcciones IP proporcionadas por un archivo de .txt "ips.txt". Este fichero se leerá por línea de comandos así como el tiempo máximo (segundos) de espera de respuesta en un salto. El script deberá calcular el tiempo medio de respuesta en el primer salto de cada destino, y luego mostrará los destinos ordenados de manera descendente basándose en este tiempo medio. Para los casos en los que en el primer salto no se obtiene respuesta en el tiempo establecido, se mostrará un mensaje de error.

Este script permitirá a los usuarios identificar cuál de los destinos proporcionados tiene el mejor tiempo de respuesta en el primer salto, lo cual puede ser un indicador de proximidad o de una conexión de red eficiente.

Hacer las comprobaciones de argumentos que considere oportunas.

Nota: las operaciones aritméticas en bash no admiten operaciones con números decimales. Para realizar este tipo de operaciones, puedes usar la herramienta bc, que resuelve una operación que esté indicada como cadena de texto. Por ejemplo: *echo "5.3 + 2.4" | bc* dará como resultado 7.7.

A continuación se muestra un ejemplo del tipo de salida esperada:

```
i72jivem@VTS3:~/PAS/p2$ ./ejercicio3.sh
Argumentos erroneos. Uso: ./ejercicio3.sh <archivo_con_IPs><timeout>

i72jivem@VTS3:~/PAS/p2$ ./ejercicio3.sh ip.txt 1
Se esperaba un fichero del tipo ips.txt

i72jivem@VTS3:~/PAS/p2$ ./ejercicio3.sh ips.txt 1
IP 192.168.100.100 ha tardado 37.78 ms en el primer salto
IP 149.112.112.112 ha tardado 12.53 ms en el primer salto
IP 208.67.222.222 ha tardado 11.31 ms en el primer salto
IP 185.228.168.9 ha tardado 11.13 ms en el primer salto
IP 192.168.1.1 ha tardado 11.09 ms en el primer salto
IP 8.8.8.8 ha tardado 10.85 ms en el primer salto
IP 10.0.0.1 ha tardado 10.58 ms en el primer salto
IP 123.123.123.123 ha tardado 10.56 ms en el primer salto

Error: No se recibi  respuesta para 234.234.234.234 en 1 segundos
```

4. `ejercicio4.sh`

Escribir un *script* que realice lo siguiente:

1. Listar todos los ficheros ocultos de la carpeta personal del usuario ordenados de menor a mayor número de caracteres
2. Listar por pantalla todos los procesos que el usuario del sistema está ejecutando en ese momento. Para cada proceso tendrá que mostrar el PID, hora en que se lanzó y nombre del fichero ejecutable. Ordénalos de forma alfabética inversa por nombre de ejecutable. (Deberás consultar información de ps) → ps -ux +tuberías

A continuación se muestra un ejemplo del tipo de salida esperada:

```
i72jivem@VTS3:~/PAS/p2$ ./ejercicio4.sh
 1) Ficheros ocultos de /home/i72jivem
.
..
.m2
.qt
.ddd
.kde
.pki
.ssh
.atom
.dbus
.dmrc
.gvfs
.java
.mcop
.adobe
.cache
.cshrc
.gconf
.hplip
.icons
.local
.login
.nedit
.pulse
.config
.gnome2
.imagej
.mcoprc
.pencil
.themes
.tomcat
.vscode
.android
.eclipse
.lesshst
.mozilla
.nemiver
.phpbrew
.esd_auth
.gimp-2.8
.nautilus
.netbeans
.evolution
.filezilla
.firestorm
.gitconfig
.gitkraken
.install4j
.wget-hsts
.ddccontrol
.subversion
.thumbnails
.Xauthority
.bash_logout
.gnuplot-wxt
.octave_hist
.python-eggs
.bash_history
.ICEauthority
.kompozer.net
```

```
.packettracer
.pulse-cookie
.vscode-server
.xmlcopyeditor
.gnome2_private
.gstreamer-0.10
.gnuplot_history
.mission-control
.rstudio-desktop
.AndroidStudio3.0
.oracle_jre_usage

----

 2)  Listado de los procesos ejecutados por el usuario  i72jivem
PID: "101990"  Hora: "12:23"  Ejecutable: "/usr/local/sbin/sftp-server"
PID: "59003"  Hora: "14:54"  Ejecutable: "sshd: i72jivem@pts/0"
PID: "101989"  Hora: "12:23"  Ejecutable: "sshd: i72jivem@notty"
PID: "103226"  Hora: "15:36"  Ejecutable: "ps ux"
PID: "103006"  Hora: "15:36"  Ejecutable: "/bin/bash ./ejercicio4.sh"
PID: "59004"  Hora: "14:54"  Ejecutable: "-bash"
```

5. `ejercicio5.sh`

Desarrollar un *script* que reciba como argumento el nombre de un fichero de texto y extraiga las palabras que se encuentren en las líneas que contienen números. Cada palabra se deberá mostrar en una línea diferente y deberán estar ordenadas por orden alfabético inverso. Además de la palabra, en cada línea se mostrará el número de orden y el número de veces que se repite la palabra (sin distinguir mayúsculas). Recuerda realizar los controles de errores oportunos.

Nota 1: para quedarte con los elementos únicos de una "lista ordenada" puedes usar el comando *uniq* mediante una tubería.

Nota 2: Para mostrar el número de orden de cada palabra puedes usar el comando *nl* mediante una tubería.

A continuación se muestra un ejemplo de salida:

```
i72jivem@VTS3:~/PAS/p2$ ./ejercicio5.sh
Argumentos incorrectos. Uso: ./ejercicio5.sh <texto.txt>

i72jivem@VTS3:~/PAS/p2$ ./ejercicio5.sh tt.txt
Se esperaba un fichero del tipo texto.txt

i72jivem@VTS3:~/PAS/p2$ ./ejercicio5.sh texto.txt
#                 Count             Word
     1                 5 y
     2                 1 veces
     3                 1 un
     4                 2 tr
     5                 2 texto
     6                 1 tail
     7                 1 su
     8                 1 sistemas
     9                 1 signos
    10                 2 sed
    11                 1 proposito
    12                 1 programacion
    13                 1 practicas
    14                 2 practica
    15                 1 posibilidades
    16                 1 pense
    17                 3 para
    18                 1 palabras
    19                 2 numeros
    20                 1 numericos
    21                 1 manipular
    22                 1 manera
    23                 1 manejo
    24                 1 las
    25                 3 la
    26                 1 incluye
    27                 1 incluso
    28                 2 grep
    29                 1 formatos
```

```
    30                  1 facilitar
    31                  2 este
    32                  1 especificamente
    33                  1 esenciales
    34                  2 es
    35                  2 en
    36                  1 el
    37                  1 ejemplos
    38                  1 eficiente
    39                  1 dise ado
    40                  1 diferentes
    41                  1 del
    42                  7 de
    43                  1 contiene
    44                  1 conceptos
    45                  2 con
    46                  1 comprension
    47                  1 complejidad
    48                  2 como
    49                  1 clase
    50                  2 bash
    51                  1 archivo
    52                  1 a o
    53                  1 a adir
    54                  1 algunos
    55                  1 administracion
    56                  1 ademas
```

6. `ejercicio6.sh`

Desarrollar un *script* que muestre el contenido del fichero */etc/passwd/* (y parte del fichero */etc/group/*) de forma amigable. El script recibirá un único argumento que será una cadena de texto que corresponde con un tipo de shell. Sólo se mostrarán los usuarios que tengan esa cadena como shell del sistema de la siguiente manera:

- Logname del usuario
- UID del usuario
- Groupname de su grupo primario (aquí tendrás que acceder a */etc/group/* sabiendo su GID)
- GID del grupo primario
- Shell por defecto

A continuación se muestra un ejemplo de salida:

```
i72jivem@VTS3:~/PAS/p2$ ./ejercicio6.sh
Argumentos incorrectos. Uso: ./ejercicio6.sh </bin/bash>

i72jivem@VTS3:~/PAS/p2$ ./ejercicio6.sh /bin/bash
=========
Logname: root
->UID: 0
->Grupo: root
->GID: 0
->Shell por defecto: /bin/bash

=========
Logname: couchdb
->UID: 106
->Grupo: couchdb
->GID: 113
->Shell por defecto: /bin/bash



i72jivem@VTS3:~/PAS/p2$ ./ejercicio6.sh /bin/sh
=========
Logname: daemon
->UID: 1
->Grupo: daemon
```

```
->GID: 1
->Shell por defecto: /bin/sh

=========
Logname: bin
->UID: 2
->Grupo: bin
->GID: 2
->Shell por defecto: /bin/sh

=========
Logname: sys
->UID: 3
->Grupo: sys
->GID: 3
->Shell por defecto: /bin/sh

=========
Logname: games
->UID: 5
->Grupo: games
->GID: 60
->Shell por defecto: /bin/sh

=========
Logname: man
->UID: 6
->Grupo: man
->GID: 12
->Shell por defecto: /bin/sh

=========
Logname: lp
->UID: 7
->Grupo: lp
->GID: 7
->Shell por defecto: /bin/sh

...


=========
Logname: irc
->UID: 39
->Grupo: irc
->GID: 39
->Shell por defecto: /bin/sh

=========
Logname: gnats
->UID: 41
->Grupo: gnats
->GID: 41
->Shell por defecto: /bin/sh

=========
Logname: nobody
->UID: 65534
->Grupo: nogroup
->GID: 65534
->Shell por defecto: /bin/sh

=========
Logname: libuuid
->UID: 100
->Grupo: libuuid
->GID: 101
->Shell por defecto: /bin/sh

=========
Logname: speech-dispatcher
->UID: 108
->Grupo: audio
->GID: 29
->Shell por defecto: /bin/sh
```