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

Convocatoria de junio (curso 2024/2025)

Pedro Antonio Gutiérrez Peña / Javier Sánchez Monedero

28 de marzo de 2025

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 pagutierrez@uco.es o jsanchezm@uco.es.

1. `ejercicio1.sh`

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

1. Años de los libros publicados entre 1950 y 2020. Los años siempre comienzan con la secuencia "Año: ".
2. Extraer y mostrar los precios que superen los 20 euros. Los precios aparecen en el formato "XX,XX€".
3. Contar cuántos libros hay por cada género, suponiendo que el género siempre está encerrado entre corchetes (por ejemplo, "[Ciencia ficción]")[1].
4. Mostrar las palabras que contengan al menos 8 caracteres que empiecen por consonante y terminen por vocal.
5. Mostrar las líneas que contengan un autor cuyo nombre o apellido tenga una doble "l". Los autores siempre comienzan con la secuencia "Autor: "
6. Títulos de libros con más de tres palabras.
7. Extraer y mostrar los títulos de libros cuyo precio termine en ",99€".
8. Contar cuántos libros tienen un año de publicación anterior a 2000.
9. Mostrar las líneas que contienen dos o más palabras con mayúscula consecutivas (sin contar la primera palabra que tiene los dos puntos, es decir, sin contar "Autor:", "Título:"...).

[1] Para este apartado, te será útil el comando `uniq -c`

10. Extraer y mostrar los géneros que contienen una palabra compuesta (separada por un guión).

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

```
i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio1.sh
Argumentos erróneos. Uso: ./ejercicio1.sh <fichero_libros>.

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio1.sh libros.xlsx
Se esperaba un fichero de tipo txt.

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio1.sh libros.txt

1) Años de los libros publicados entre 1950 y 2020:
2014
2015
2012
2007
2011
2016
2020
2012
1951
1965
1967
2003
1989
2002

2) Precios superiores a 20 euros:
24,50€
22,75€
35,99€
21,00€
25,99€
27,99€
23,75€
20,99€

3) Número de libros por género:
Auto-ayuda aparece 1 veces
Ciencia-ficción aparece 4 veces
Distopía aparece 1 veces
Fantasía aparece 2 veces
Historia aparece 1 veces
Misterio aparece 3 veces
Novela histórica aparece 1 veces
Realismo mágico aparece 1 veces
Thriller aparece 1 veces
Thriller psicológico aparece 1 veces

4) Palabras de al menos 8 caracteres que empiezan por consonante y terminan por vocal:
problema
Misterio
Historia
Historia
Fantasía
marciano
silencio
Misterio
Misterio
Distopía
Realismo
histórica
psicoanalista
psicológico
Fantasía

5) Líneas de autor cuyo nombre o apellido contiene una doble "l":
Autor: George Orwell
Autor: Ken Follett

6) Títulos de libros con más de tres palabras:
Título: El problema de los tres cuerpos
Título: La chica del tren
Título: Historia mínima de España
Título: El nombre del viento
Título: El silencio de la ciudad blanca
Título: Los secretos que jamás te contaron
```

```
Título: La verdad sobre el caso Harry Quebert
Título: Cien años de soledad
Título: El código Da Vinci
Título: Los pilares de la Tierra

7) Títulos de libros cuyo precio termina en ,99€:
Título: Historia mínima de España
Título: El nombre del viento
Título: Fundación
Título: Dune
Título: El código Da Vinci
Título: El psicoanalista
Título: El Hobbit

8) Número de libros con año de publicación anterior a 2000:
6

9) Líneas con dos o más palabras con mayúscula consecutivas:
Autor: Cixin Liu
Autor: Paula Hawkins
Autor: Juan Pablo Fusi
Autor: Patrick Rothfuss
Autor: Andy Weir
Autor: Eva García Sáenz de Urturi
Autor: Albert Espinosa
Título: La verdad sobre el caso Harry Quebert
Autor: Joël Dicker
Autor: Isaac Asimov
Autor: George Orwell
Autor: Frank Herbert
Autor: Gabriel García Márquez
Título: El código Da Vinci
Autor: Dan Brown
Autor: Ken Follett
Autor: John Katzenbach
Título: El Hobbit

10) Géneros con una palabra compuesta:
[Género: Auto-ayuda]
[Género: Ciencia-ficción]
```

2. `ejercicio2.sh`

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

```
Título: XXX
| -> Autor: XXX
| -> Año: XXX
| -> Precio: XXX
| -> Género: XXX
```

Además, el script deberá mostrar la salida por terminal.

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

```
i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio2.sh
Argumentos erróneos . Uso: ./ejercicio2.sh <fichero_libros>.

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio2.sh libros.xlsx
Se esperaba un fichero del tipo txt

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio2.sh libros.txt
Título: El problema de los tres cuerpos
| -> Autor: Cixin Liu
| -> Año: 2014
| -> Precio: 24,50€
| -> Género: Ciencia-ficción
Título: La chica del tren
| -> Autor: Paula Hawkins
| -> Año: 2015
| -> Precio: 22,75€
```

```
| -> Género: Misterio
Título: Historia mínima de España
| -> Autor: Juan Pablo Fusi
| -> Año: 2012
| -> Precio: 19,99€
| -> Género: Historia
Título: El nombre del viento
| -> Autor: Patrick Rothfuss
| -> Año: 2007
| -> Precio: 35,99€
| -> Género: Fantasía
Título: El marciano
| -> Autor: Andy Weir
| -> Año: 2011
| -> Precio: 18,00€
| -> Género: Ciencia-ficción
Título: El silencio de la ciudad blanca
| -> Autor: Eva García Sáenz de Urturi
| -> Año: 2016
| -> Precio: 16,50€
| -> Género: Misterio
Título: Los secretos que jamás te contaron
| -> Autor: Albert Espinosa
| -> Año: 2020
| -> Precio: 14,95€
| -> Género: Auto-ayuda
Título: La verdad sobre el caso Harry Quebert
| -> Autor: Joël Dicker
| -> Año: 2012
| -> Precio: 21,00€
| -> Género: Misterio
Título: Fundación
| -> Autor: Isaac Asimov
| -> Año: 1951
| -> Precio: 25,99€
| -> Género: Ciencia-ficción
Título: 1984
| -> Autor: George Orwell
| -> Año: 1949
| -> Precio: 17,50€
| -> Género: Distopía
Título: Dune
| -> Autor: Frank Herbert
| -> Año: 1965
| -> Precio: 27,99€
| -> Género: Ciencia-ficción
Título: Cien años de soledad
| -> Autor: Gabriel García Márquez
| -> Año: 1967
| -> Precio: 23,75€
| -> Género: Realismo mágico
Título: El código Da Vinci
| -> Autor: Dan Brown
| -> Año: 2003
| -> Precio: 20,99€
| -> Género: Thriller
Título: Los pilares de la Tierra
| -> Autor: Ken Follett
| -> Año: 1989
| -> Precio: 19,50€
| -> Género: Novela histórica
Título: El psicoanalista
| -> Autor: John Katzenbach
| -> Año: 2002
| -> Precio: 18,99€
| -> Género: Thriller psicológico
Título: El Hobbit
| -> Autor: J.R.R. Tolkien
| -> Año: 1937
| -> Precio: 15,99€
| -> Género: Fantasía
```

3. `ejercicio3.sh`

Este ejercicio consiste en utilizar `sed` y `grep` para analizar un archivo de acceso web de un servidor Apache, como viene en el fichero `access.log`.

El archivo de acceso contiene registros de solicitudes HTTP a un servidor Apache, y cada línea sigue el formato estándar de acceso de Apache, que incluye la dirección IP, fecha y hora, tipo de solicitud, URL solicitada, código de respuesta HTTP, y tamaño de la respuesta en bytes.

El objetivo de este ejercicio es crear un *script* en `bash` utilizando `sed` y `grep` que realice lo siguiente:

- Filtrar las líneas que contienen solicitudes exitosas (código de respuesta `HTTP 200`).
- Extraer solo la dirección IP y la URL solicitada de esas líneas.
- Ordenar las solicitudes por dirección IP y mostrar el resultado, repitiendo las líneas cuando una misma IP pide el mismo recurso en distintas fechas.
- Opcionalmente (estableciendo `repite-no` en lugar de `repite-si`), eliminar las líneas que contienen solicitudes de un mismo cliente repetidas, dejando solo la primera aparición.

El *script* debe ejecutarse de la siguiente manera:

```
./ejercicio3.sh archivo_de_acceso.log [repite-si,repite-no]
```

donde `archivo_de_acceso.log` es el archivo que contiene los registros de acceso del servidor Apache. Si el *script* se invoca con `repite-si` entonces se repetirán los accesos de la misma IP, si se invoca con `repite-no` entonces no se repetirán y se incluirá el número total de accesos.

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

```
i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio3.sh
Por favor, proporciona el archivo de acceso.

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio3.sh access.log
Por favor, proporciona la opción 'repite-si' o 'repite-no'.

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio3.sh access.log repiit
Modo no válido. Usa 'repite-si' o 'repite-no'.

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio3.sh access.log repite-si
192.168.1.1 /about
192.168.1.1 /about
192.168.1.1 /contact
192.168.1.1 /contact
192.168.1.1 /index.html
192.168.1.1 /index.html
192.168.1.1 /index.html
192.168.1.1 /login
192.168.1.1 /products
192.168.1.1 /signup
192.168.1.10 /about
192.168.1.10 /home
192.168.1.11 /home
192.168.1.12 /about
192.168.1.13 /login
192.168.1.14 /index.html
192.168.1.2 /login
192.168.1.2 /signup
192.168.1.3 /index.html
192.168.1.3 /index.html
192.168.1.4 /about
192.168.1.4 /products
192.168.1.5 /login
192.168.1.6 /about
192.168.1.6 /home
192.168.1.7 /login
192.168.1.7 /products
192.168.1.7 /signup
192.168.1.8 /index.html
192.168.1.9 /login

i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio3.sh access.log repite-no
192.168.1.1 /about --> 2 veces
192.168.1.1 /contact --> 2 veces
192.168.1.1 /index.html --> 3 veces
192.168.1.1 /login --> 1 veces
192.168.1.1 /products --> 1 veces
192.168.1.1 /signup --> 1 veces
```

```
192.168.1.10 /about --> 1 veces
192.168.1.10 /home --> 1 veces
192.168.1.11 /home --> 1 veces
192.168.1.12 /about --> 1 veces
192.168.1.13 /login --> 1 veces
192.168.1.14 /index.html --> 1 veces
192.168.1.2 /login --> 1 veces
192.168.1.2 /signup --> 1 veces
192.168.1.3 /index.html --> 2 veces
192.168.1.4 /about --> 1 veces
192.168.1.4 /products --> 1 veces
192.168.1.5 /login --> 1 veces
192.168.1.6 /about --> 1 veces
192.168.1.6 /home --> 1 veces
192.168.1.7 /login --> 1 veces
192.168.1.7 /products --> 1 veces
192.168.1.7 /signup --> 1 veces
192.168.1.8 /index.html --> 1 veces
192.168.1.9 /login --> 1 veces
```

4. `ejercicio4.sh`

Desarrolla un script que consulte el archivo `/etc/passwd` y realice las siguientes tareas:

1. Mostrar todos los usuarios cuyo nombre empieza con la letra "`l`".
2. Mostrar los usuarios cuyo *shell* es válido (es decir, diferente de `/bin/false` o `/usr/bin/nologin`).
3. Mostrar el UID de los usuarios cuyo directorio `home` no está en `/home`.
4. Mostrar los usuarios que tienen asignado un GID mayor que 1000.
5. Mostrar los usuarios y su UID con una "`,`" en su `gecos`.

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

```
1. Usuarios cuyo nombre empieza con la letra 'l':
lp
list
libuuid

2. Usuarios con shell válido:
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
libuuid
couchdb
speech-dispatcher
sshd
nx

3. UID de los usuarios cuyo directorio home no está en home:
0
1
2
3
4
5
```

```
6
7
8
9
10
13
33
34
38
39
41
65534
100
98
103
104
85
106
87
108
109
85
97
113
114
90
117
9545

4. Usuarios con GID mayor que 1000:
sync
nobody
kernoops
sshd
nx

5. Usuarios y su UID con una ',' en su gecos:
hplip, 103
avahi-autoipd, 104
avahi, 85
couchdb, 106
haldaemon, 87
speech-dispatcher, 108
kernoops, 109
pulse, 85
mysql, 114
polkituser, 90
usbmux, 116
rtkit, 117
```

5. `ejercicio5.sh`

Utilizar el comando `ifconfig` para mostrar las interfaces de red disponibles. En base a su resultado, para cada interfaz, queremos extraer:

1. La dirección IP que tiene asignada.
2. La dirección de *broadcast* y la máscara de red. Para la máscara de red, queremos obtener el número de unos a la izquierda de la IP que estamos utilizando (típicamente, 8, 16 o 24, pero debes hacerlo convirtiendo la máscara de red[2]).
3. Comprobar si tiene el cable conectado. Para ello, utilizar la herramienta `ethtool` y comprobar si la opción "`Link detected`" aparece como afirmativa.
4. De nuevo, utilizando `ethtool`, enumerar las velocidades a las que puede trabajar la interfaz y cuáles de ellas están anunciadas como disponibles.

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

[2]Para esta tarea, puedes usar el comando `echo ''obase=2; ibase=10; $octet''| bc`, que convierte de decimal a binario lo que haya en la variable `octet`

```
1
2  i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio5.sh
3  Interfaz: eth0
4    -> Dirección IP: 172.26.2.200
5    -> Broadcast: 172.26.2.255
6    -> Máscara de red: 24 bits
7    -> Cable conectado: Sí
8    -> Velocidades soportadas: 10baseT/Half 10baseT/Full  100baseT/Half 100baseT/Full  1000baseT/Full
9    -> Velocidades anunciadas como disponibles: 10baseT/Half 10baseT/Full  100baseT/Half 100baseT/Full
         1000baseT/Full
10
11 Interfaz: lo
12   -> Dirección IP: 127.0.0.1
13   -> Broadcast: 0.0.0.0
14   -> Máscara de red: 8 bits
15   -> Cable conectado: Sí
16   -> Velocidades soportadas: No disponible
17   -> Velocidades anunciadas como disponibles: No disponible
```

6. `ejercicio6.sh`

Este ejercicio consiste en analizar los últimos 20 comandos ejecutados por el usuario en la terminal utilizando el historial de comandos (`.bash_history`). A partir de estos comandos, el *script* debe proporcionar la siguiente información:

1. Contar cuántas veces ha sido ejecutado cada comando.
2. Determinar cuál ha sido el número máximo de argumentos con los que se ha invocado cada comando.

NOTA1: para los efectos del ejercicio, no se van a considerar tuberías o el separador ";". Es decir, el comando de cada línea siempre será la primera palabra y, si hay tuberías, se considera que son argumentos al comando detectado al principio de la línea. P.ej.: `echo 2 | tail -c 4` contabiliza como un comando `echo` con 5 argumentos.
NOTA2: El propio nombre del comando **no** se contabiliza como argumento.

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

```
1  i02gupep@VTS1:~/pas/2425/p2$ cat ~/.bash_history | tail -n 20
2  cd
3  cd pas/2425/
4  ln /etc ./prueba
5  ln /etc ./prueba -s
6  rm ./prueba
7  echo $HOME
8  echo "Proban" "Prueba" "Prueba" "Prueba"
9  echo "Proban" "Prueba" "Prueba"
10 echo "Proban" "Prueba"
11 echo "Proban"
12 echo 2
13 echo 2 | tail -c 1
14 echo 2 | tail -c 2
15 echo 2 | tail -c 4
16 cat $HOME/meloinvento
17 cat $HOME/.bash_history | tail -n 20
18 ls
19 cd p2/
20 ./ejercicio6.sh
21 exit
22
23 i02gupep@VTS1:~/pas/2425/p2$ ./ejercicio6.sh
24 Comando: cat
25   -> Veces ejecutado: 2
26   -> Máximo número de argumentos: 5
27
28 Comando: cd
29   -> Veces ejecutado: 3
30   -> Máximo número de argumentos: 1
31
32 Comando: echo
33   -> Veces ejecutado: 9
```

```
  -> Máximo número de argumentos: 5

Comando: ./ejercicio6.sh
  -> Veces ejecutado: 1
  -> Máximo número de argumentos: 1

Comando: exit
  -> Veces ejecutado: 1
  -> Máximo número de argumentos:

Comando: ln
  -> Veces ejecutado: 2
  -> Máximo número de argumentos: 3

Comando: ls
  -> Veces ejecutado: 1
  -> Máximo número de argumentos:

Comando: rm
  -> Veces ejecutado: 1
  -> Máximo número de argumentos: 1
```