




LABORATORIO DE SISTEMAS (GIRS)

PRÁCTICA 1 - SHELL SCRIPTING

23 DE FEBRERO DE 2023

 **Profesor:** Felipe Ortega.

 **Versión:** 1.0.

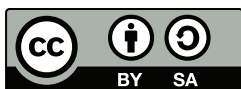
 **Lenguaje de programación:** Bash, sed, awk, Python.

 **Fecha tope de entrega:** Domingo, 5 de marzo de 2023.

 **Envío:** Código fuente **comentado**, a través del espacio de entrega en Aula Virtual.

OBJETIVOS

- Afianzar la programación de *scripts* para la Shell.
- Reforzar los conocimientos sobre herramientas importantes como sed y awk.
- Consolidar la programación de *scripts* en Python.



© 2023 Elena García-Morato, Felipe Ortega.
Algunos derechos reservados.

PUBLICADO POR GSyC – ETSIT

<https://www.urjc.es/etsit>

Este trabajo se entrega bajo la licencia Creative Commons
Reconocimiento-Compartir-Igual (BY-SA). Para obtener
la licencia completa, véase:
<https://creativecommons.org/licenses/by-sa/4.0/deed.es>.

I SHELL SCRIPTING

Ejercicio 1

Escribe un *script* de shell llamado `filebysize.sh` que se comporte de la siguiente manera:

- ▶ Si se ejecuta el *script* sin ninguna opción, deberá imprimir por pantalla el nombre del fichero de mayor tamaño del directorio actual (sin búsqueda recursiva).
- ▶ Si se ejecuta con la opción `-s`, deberá imprimir por pantalla el nombre del fichero de menor tamaño del directorio actual (sin búsqueda recursiva).
- ▶ Si se ejecuta con la opción `-r`, deberá imprimir por pantalla el nombre del fichero de mayor tamaño en el directorio actual o en alguno de sus subdirectorios (búsqueda recursiva).
- ▶ Si se ejecuta con las opciones `-r -s`, deberá imprimir por pantalla el nombre del fichero de menor tamaño en el directorio actual o en alguno de sus subdirectorios (búsqueda recursiva).
- ▶ Si recibe cualquier otra opción como parámetro o un número incorrecto de parámetros debe imprimir un mensaje de error alertando sobre ello y finalizar con el [código de status apropiado](#)¹.

¹ Devuelve un código diferente para cada tipo de error

Ejercicio 2

Escribe un *script* de shell llamado `processes.sh` que se comporte de la siguiente forma:

- ▶ Obligatoriamente debe recibir al menos un parámetro de entrada, que represente el nombre de un usuario del sistema. En caso de que este parámetro no se introduzca, debe imprimir un mensaje de error alertando sobre ello y finalizar con el [código de status apropiado](#).
- ▶ Si se ejecuta con la opción `-f`, deberá imprimir por pantalla el nombre del usuario introducido y, a continuación, el número de procesos de ese usuario que están actualmente en ejecución en primer plano.

- ▶ Si se ejecuta con la opción `-b`, deberá imprimir por pantalla el nombre del usuario introducido y, a continuación, el número de procesos de ese usuario que están actualmente en ejecución en segundo plano.
- ▶ Si se ejecuta con la opción `-a` o no se pasa ningún otro parámetro además del nombre del usuario deberá imprimir por pantalla el nombre del usuario introducido junto con el número total de procesos de ese usuario.
- ▶ Si recibe cualquier otra opción como parámetro o un número incorrecto de parámetros debe imprimir un mensaje de error alertando sobre ello y finalizar con el [código de status apropiado](#)².

² Devuelve un código diferente para cada tipo de error

Ejercicio 3

El fichero `/etc/hosts` contiene una lista de direcciones IP y nombres de máquinas asociadas a cada IP. A continuación se muestra un extracto para una de las máquinas de Laboratorios ETSIT.

```
# L.2.108

212.128.254.2 f-l2108-pc00.aulas.etsit.urjc.es f-l2108-pc00
212.128.254.3 f-l2108-pc01.aulas.etsit.urjc.es f-l2108-pc01
212.128.254.4 f-l2108-pc02.aulas.etsit.urjc.es f-l2108-pc02
212.128.254.5 f-l2108-pc03.aulas.etsit.urjc.es f-l2108-pc03
212.128.254.6 f-l2108-pc04.aulas.etsit.urjc.es f-l2108-pc04
212.128.254.7 f-l2108-pc05.aulas.etsit.urjc.es f-l2108-pc05
212.128.254.8 f-l2108-pc06.aulas.etsit.urjc.es f-l2108-pc06
212.128.254.9 f-l2108-pc07.aulas.etsit.urjc.es f-l2108-pc07
212.128.254.10 f-l2108-pc08.aulas.etsit.urjc.es f-l2108-pc08

...
```

Escribe un *script* de shell llamado `idhosts.sh` con el siguiente comportamiento:

- ▶ Si no recibe ningún argumento, imprime por pantalla el número total de nombres de máquinas que aparecen en el fichero `/etc/hosts`.
- ▶ Si recibe el parámetro `-l` seguido del nombre de un laboratorio, imprime por pantalla el número total de nombres de máquinas de ese laboratorio. Por ejemplo:

```
$ ./idhosts.sh -l L.2.108
L.2.108 has 19 hosts
```

- ▶ Si recibe el parámetro `-l` seguido del nombre de un laboratorio y también el parámetro `-n` entonces debe imprimir por pantalla una lista con los números de PC de todas las máquinas de ese laboratorio. Por ejemplo:

```
$ ./idhosts.sh -l L.2.108 -n
L.2.108: pc00 pc01 pc02 ... [resto de identificadores]
```

- ▶ Si recibe cualquier otra opción como parámetro o un número incorrecto de parámetros debe imprimir un mensaje de error alertando sobre ello y finalizar con el [código de status apropiado](#)³.
- ▶ Si recibe un identificador de laboratorio que no existe en el archivo debe imprimir un mensaje de error avisando sobre esto y finalizar con el [código de status apropiado](#).

³ Devuelve un código diferente para cada tipo de error

Ejercicio 4

Modifica el *script* anterior, creando uno nuevo llamado `idhosts-new.sh`. El nuevo *script* debe imprimir los resultados de cada apartado con un formato más completo.

- ▶ Si no recibe ningún argumento, imprime por pantalla el número total de nombres de máquinas que aparecen en el fichero `/etc/hosts`.
- ▶ Si recibe el parámetro `-l` seguido del nombre de un laboratorio, imprime por pantalla:

```
$ ./idhosts.sh -l L.2.108

Lab: L.2.108

Total hosts: 19

First host: f-l2108-pc00 | 212.128.254.2

Last host: f-l2108-pc18 | 212.128.254.20
```

- ▶ Si recibe el parámetro `-l` seguido del nombre de un laboratorio y también el parámetro `-n` entonces debe imprimir:

```
$ ./idhosts.sh -l L.2.108 -n
```

Lab: L.2.108

Id PC	IP	Subdomain
-----	-----	-----
pc00	212.128.254.2	aulas.etsit.urjc.es
pc01	212.128.254.3	aulas.etsit.urjc.es
pc02	212.128.254.4	aulas.etsit.urjc.es
pc03	212.128.254.5	aulas.etsit.urjc.es
...		
[resto de filas]		

Total hosts: 19

II PYTHON SCRIPTING

El archivo `Pima.csv` se ha publicado junto al enunciado de esta práctica en Aula Virtual.

Descarga el archivo en tu máquina local para poder completar los siguientes ejercicios de creación de *scripts* en Python.

Ejercicio 5

Crea un *script* en Python llamado `pima_res.py` que se ajuste a las siguientes especificaciones:

- ▶ Como mínimo, debe recibir como argumento de entrada posicional obligatorio el nombre del archivo CSV a procesar. En este caso es `Pima.csv`.
- ▶ Si recibe el parámetro `-h` o bien `--help` imprime información de ayuda sobre la sintaxis para ejecutar el *script* y las opciones disponibles.
- ▶ Si recibe el parámetro `-m` o bien `--mean` imprime el nombre de cada columna que contenga valores numéricos, junto con el promedio de los datos de esa columna.

- ▶ Si recibe el parámetro `-b` o bien `--bool` imprime el nombre de cada columna que contenga valores booleanos (no solo `True` o `False`, también pueden codificarse como `Yes/No`), junto con el número total de filas con valor `True` para esa columna y el número total de filas con valor `False` para esa columna ⁴.

⁴ Consideramos que `Yes` equivale a `True` y `No` equivale a `False`.