Exámenes Resueltos

Informática

magister

EJERCICIOS DE EXAMEN DE OPOSICIONES SOBRE SCRIPT LINUS-1:

ARAGÓN INFORMÁTICA- 2018

PARTE A- EJERCICIO 2:

3.6.- Expresiones Regulares

Teniendo en cuenta la salida proporcionada por el siguiente comando:

```
$ ls -1 /var/mislog
...
2015-1-13-estado_hardware.log.1
modificado-2015-1-13-estado_hardware.log.1
2015-1-21-estado_hardware.Log
2015-1-31-estado_hardware.log.gz
2015-1-6-estado_hardware.log.2
2015-1-7-estado_servicios.lg
2015-1-7-estado_software.audit
2015-1-31-estado_hardware.log.1.zip
...
2015-2-5-auditoria_impresion.log.3
modificado-2015-2-5-auditoria_impresion.log.3
...
2017-12-31-auditoria_anual.tar.gz
...
rectificado-2018-6-23-estado_del_arranque.LOG
2018-6-23-estado_del_arranque.LOG
```

(A) Se desea filtrar el listado de archivos resultante mediante el comando "egrep" haciendo uso de una expresión regular estándar, de tal forma que cumpla los siguientes requisitos:

```
$ ls -1 /var/mislog | egrep "expresión_regular"
```

- Se deberán mostrar los archivos de log de los 4 primeros meses del año 2015.
- El listado resultante deberá omitir aquellos archivos que no comiencen directamente el año 2015, omitiéndose los "modificados", "rectificados", etc.
- Deberán listarse únicamente los archivos de auditoria no comprimidos o empaquetados terminados en alguna de las siguientes extensiones: ".lg", ".log", ".log.1", ".log.2", ".audit", etc. pudiendo estar estas extensiones indistintamente en mayúsculas o minúsculas (*Log, log, LOG, LG, lg, audit, Audit, ...*)

(B) Haciendo uso del comando anterior escribe un pequeño script de sistema en bash que se encargará de eliminar de los ficheros de auditoria resultantes del filtrado aquellas líneas que contengan un patrón de búsqueda. En concreto el script deberá recibir como parámetros los siguientes datos:

```
$ miscript 2015 3 9 "system initialization"
```

- Primer parámetro: año de los archivos de log deseados.
- Segundo y tercer parámetro: el intervalo de meses.
- Cuarto parámetro: la cadena de texto o patrón de búsqueda que permitirá decidir que líneas serán eliminadas de los ficheros de auditoría resultantes.
- En el caso de que el número de parámetros sea diferente de cuatro se deberá mostrar un mensaje de error advirtiendo de la necesidad de que sean 4 los parámetros a enviar.

Valor: 0,25 (A)+ 0,25 (B) = 0,5 puntos

SOLUCIÓN:

(A)

1. Se pide extraer los archivos de los cuatro primeros meses del año 2015, es decir, viendo los nombres de los ficheros, buscamos los aquellos que empiezan por 2015, a continuación tienen un guión, después un número entre 1 y 4 y después cualquier cosa. La expresión regular será:

egrep ^2015-[1-4]

2. egrep ^2015

3. egrep auditoria.+(log|lg|log.1|… | log.3)$

(B)

```
#!/bin/bash
# Primero extraemos los ficheros, por ejemplo los pedidos en el primer ejemplo
(ls -1 | egrep ^2015-[1-4].+\.[1-3]$) > fichero1.txt
#Evaluamos cuántos parámetros llegan
if [ $# = 4 ]
then
# Ahora hay que leer línea a línea cada fichero
        while read line; do
                echo -e "Nombre de fichero: $line";
                cat $(echo -e "$line\n") | egrep ^$1-[$2-$3].+$4.+ ;
                echo "-----------------------";
                echo "-------------------";
        done < fichero1.txt
else
        echo "Error el número de parámetros debe ser 4"
fi
```

MADRID INORMÁTICA-2018

Ejercicio 1 (2,5 ptos)

Escriba un script llamado ejercicio1.sh que imprima un listado de los números primos en el intervalo [A,B]. A y B serán pasados como parámetros. Ejemplo de llamada:

$ ./ejercicio1.sh A B

Ejemplo de ejecución y salida generada para el intervalo [5,14]:

$ ./ejercicio1.sh 5 14
7
11

13

NOTA: Puede comprobar si un número es primo analizando la salida de la utilidad factor incluida en el paquete GNU coreutils (que suponemos estará instalado en su sistema). Dicha utilidad factoriza números enteros. Ejemplo de uso y salida generada de la llamada a la utilidad factor:

$ factor 2018 2018: 2 1009

SOLUCIÓN:

```
#!/bin/bash
# leemos el intervalo
fin=$(($2 + 1))

echo 'Entre ' $1 ' y ' $2 ' hay estos números primos:'
echo " "
# Recorremos desde el primer número hasta el último
# Usamos dos ficheros de apoyo para guardar el resultado de factor
# Utilizamos el comando cut para extraer el contenido de los ficheros
# y comparar la salida del comando.
for ((i=$1;i<$fin;i++))
do
        factor $i > fichApoyo
        uno=$(cut -d ':' -f 1 fichApoyo)
        cut -d ':' -f 2 fichApoyo > fichApoyo2
        dos=$(cut -d ' ' -f 2 fichApoyo2)

        if [[ $uno -eq $dos ]]
        then
                echo $uno
#       else
#               echo "distinto"
        fi
done
rm fichApoyo
rm fichApoyo2
```

LA RIOJA 2018

El siguiente ejercicio tiene una puntuación de 1,5 puntos

El administrador de sistemas para controlar la información de los trabajadores de la empresa LARSAI, decide realizar un **script bash parametrizado** llamado "**informa**" en el que no se solicitará información al usuario durante su ejecución, ya que toda la información necesaria se incluirá en forma de parámetros. Concretando más, se tratará de manipular la información contenida en el siguiente fichero llamado "**bdtrabajadores**" (base datos de los trabajadores de la empresa), que tiene el siguiente formato:

```
N°Empleado:Nombre:Apellido:DNI:SueldoMensual(euros):Puesto
100:Neo:Elegido:16588099D:1660:Director
101:Trinity:Alonso:16652899G:1380:Subdirector
102:Niobe:Epifanio:23422161L:966:Diseñador
103:Morfeo:Navarro:33288715H:1200:Programador
104:Persefone:Gasol:56376544K:2100:Tecnico
```

Las opciones que acepta el script son: -m, -a y –g, y se detallan a continuación:

-m → Mostrará la información del empleado que se le indique a través de su DNI *(0,5 puntos)*

A continuación se muestra un ejemplo donde se aprecia la ejecución y el formato de salida en pantalla.

```
[usuario@Linux] informa -m 33288715H
N°Empleado Nombre Apellido DNI SueldoMensual(euros) Puesto
103 Morfeo Navarro 33288715H 1200 Programador
```

-a → Sueldo anual *(0,5 puntos)*. Informará del sueldo anual que percibirá un determinado empleado, el cual se especificará igualmente haciendo uso de su DNI. Para llegar a determinar el sueldo anual se tendrá que tener en cuenta las dos extraordinarias, la de junio y la de diciembre, donde el valor de esta depende del puesto de trabajo del colaborador: Técnico (1500 euros de extraordinaria), Director (900 euros de extraordinaria), Subdirector (854 euros de extraordinaria), Programador (700 euros) y Diseñador (500 euros).

Un ejemplo sería el siguiente:

```
[usuario@Linux] informa -a 56376544K
Nombre Apellido DNI Sueldo Mensual Sueldo Anual
Persefone Gasol 56376544K 2100 euros 28200 euros
```

Nota: 2100x12 + 1500x2 = 28200 €, 12 meses a 2100 € por mes, 2 extraordinarias de 1500 € al año

-g → Guardar sueldo anual *(0,5 puntos)*. Generará un nuevo fichero de texto pasado como parámetro donde guardará toda la información contenida en el fichero bdtrabajadores del primer apartado pero añadiendo la información del sueldo anual del segundo apartado:

```
[usuario@Linux] informa -g sueldos.txt
N°Empleado:Nombre:Apellido:DNI:SueldoMensual(euros):Puesto:SueldoAnual
100:Neo:Elegido:16588099D:1660:Director:21720
...
104:Persefone:Gasol:56376544K:2100:Tecnico:28200
```

SOLUCIÓN:

```
#!/bin/bash

# Parámetro -m: mostrar la línea del fichero correspondiente al DNI que se pasa  # como parámetro
echo "Ejecute el script con los siguientes parámetros: "
echo "-m + DNI Si quiere saber los datos de un usuario"
echo "-a + DNI para saber el sueldo anual"
echo "-g + nombre fichero si lo que quiere es guardar un fichero  añadiendo el sueldo anual"
echo " "
# Utilizamos un case para hacer cosas distintas según se pase un parámetro u otro

case $1 in
"-m")
# Leemos el fichero y pasamos los datos a variables. Extraemos con el grep el DNI  # que buscamos y lo pasamos
# a un fichero de apoyo para luego extraer las distintas columnas.
        more bdtrabajadores | grep $2 > fichApoyo
        if [[ -s fichApoyo ]]
        then

                nEmpleado=$(cut -d ":" -f 1 fichApoyo)
                nombre=$(cut -d ":" -f 2 fichApoyo)
                apellidos=$(cut -d ":" -f 3 fichApoyo)
                dni=$(cut -d ":" -f 4 fichApoyo)
                sueldo=$(cut -d ":" -f 5 fichApoyo)
                puesto=$(cut -d ":" -f 6 fichApoyo)
                echo "NºEmpleado Nombre Apellido DNI Sueldo Puesto"
                echo $nEmpleado $nombre $apellidos $dni $sueldo $puesto

        else

                echo "No hay datos para ese DNI"

        fi
        ;;
"-a")
# Operamos de manera similar al punto anterior
# solo que añadimos el campo calculado
        more bdtrabajadores | grep $2 > fichApoyo
        if [[ -s fichApoyo ]]
        then

                nombre=$(cut -d ":" -f 2 fichApoyo)
                apellidos=$(cut -d ":" -f 3 fichApoyo)
                dni=$(cut -d ":" -f 4 fichApoyo)

                sueldo=$(cut -d ":" -f 5 fichApoyo)

                puesto=$(cut -d ":" -f 6 fichApoyo)
                case $puesto in
                        "Tecnico")
```

```
                        sueldoAn=$((12*sueldo + 1500 + 1500))
                        ;;
                "Director")
                        sueldoAn=$((12*sueldo + 900 + 900))
                        ;;
                "Subdirector")
                        sueldoAn=$((12*sueldo + 854 + 854))
                        ;;
                "Programador")
                        sueldoAn=$((12*sueldo + 700 + 700))
                        ;;
                "Diseñador")
                        sueldoAn=$((12*sueldo + 500 + 500))
                        ;;
        esac


        echo "Nombre Apellido DNI Sueldo mensual Sueldo Anual"
        echo $nombre $apellidos $dni $sueldo $sueldoAn "euros"

else

echo "No hay datos para ese DNI"

fi
;;
"-g")
# Verificamos primero si hay tres parámetros, si los hay pasamos a leer línea a  # línea del fichero
# Aquí en vez de usar un fichero de apoyo usamos una variable, así vemos otra    # forma de hacerlo.
# Además en este caso tenemos que tratar todo el fichero y no solo una línea.
if [[ $# = 2 ]]
then
for linea in $(cat bdtrabajadores)
do
        nEmpleado=$(echo $linea | cut -d ":" -f 1)
        nombre=$(echo $linea | cut -d ":" -f 2)
        apellidos=$(echo $linea | cut -d ":" -f 3)
        dni=$(echo $linea | cut -d ":" -f 4)
        sueldo=$(echo $linea | cut -d ":" -f 5)
        puesto=$(echo $linea | cut -d ":" -f 6)

        case $puesto in
                "Tecnico")
                        sueldoAn=$((12*sueldo + 1500 + 1500))
                        echo
$nombre":"$apellidos":"$dni":"$sueldo":"$puesto":"$sueldoAn >> $2
                        ;;
                "Director")
                        sueldoAn=$((12*sueldo + 900 + 900))
```

```
                                echo
$nombre":"$apellidos":"$dni":"$sueldo":"$puesto":"$sueldoAn >> $2
                                ;;
                        "Subdirector")
                                sueldoAn=$((12*sueldo + 854 + 854))
                                echo
$nombre":"$apellidos":"$dni":"$sueldo":"$puesto":"$sueldoAn >> $2
                                 ;;
                        "Programador")
                                sueldoAn=$((12*sueldo + 700 + 700))
                                echo
$nombre":"$apellidos":"$dni":"$sueldo":"$puesto":"$sueldoAn >> $2
                                ;;
                        "Diseñador")
                                sueldoAn=$((12*sueldo + 500 + 500))
                                echo
$nombre":"$apellidos":"$dni":"$sueldo":"$puesto":"$sueldoAn >> $2
                                ;;
                        *)
                                echo    "Nombre:Apellido:DNI:Sueldo    mensual:Puesto:Sueldo
Anual" >> $2
                esac
        done
                echo "Fichero " $2 " creado correctamente"
                cat $2
        else
                echo "Parámetros incorrectos"
        fi
        ;;
*)
        echo "No ha elegido una opción válida"
        ;;
Esac
```

EXTREMADURA 2018 SAI

SCRIPT Linux:

1. Confecciona un script "creafich" que:

1. Pedirá por teclado un nombre de fichero. (Valoración 1 punto)

2. Comprobará si existe o no. En caso afirmativo mostrará el mensaje: “El fichero ya existe, desea sobreescribirlo”. Si se responde que “no”, se saldrá del proceso. (Valoración 1 punto)

3. Tanto si se responde que “sí” como si no existe, se pedirá por teclado nombre y apellidos de los alumnos y nota (será un entero entre 1 y 10) y se creará el fichero con la estructura siguiente: (Valoración 1 punto)

Nombre y Apellidos: Nota

4. Se introducirán alumnos hasta que digamos que no queremos introducir más

alumnos. (Valoración 1 punto)

5. Este fichero se pasará corno parámetro al siguiente script. (Valoración 1 punto)

2. Crea un script (estadística) que:

1. Compruebe que se pasa exactamente un parámetro. (Valoración 1 punto)

2. Compruebe que se trata de un fichero existente y que no está vacío. (Valoración

2 puntos)

3. Lea línea a línea el fichero pasado como parámetro y muestre una salida similar a la siguiente: (Valoración 2 puntos)

Total alumnos: XX

Calificación	Número	Porcentaje
1	XX	XX%
2	XX	XX%
--	XX	XX%
10	XX	XX%
Totales	XX	XX%

Alumnos con un 10

xxxxxxxxxxxxxxxxx

.....

Alumnos con un 1

Xxxxxxxxxxxxxxxxx

..…

SOLUCIÓN PRIMER SCRIPT

```
#!/bin/bash
# Hacemos las comprobaciones que nos piden, utilizamos funciones.

function pedirDatos()
{
        read -p "Escribe el nombre " nombre
        read -p "Apellidos " apellidos
        read -p "y nota (1-10) " nota
}
```

```
function seguimos()
{
	read -p "¿Quieres introducir un nuevo alumno? y/n **** "  respuesta
}

if [[ $# -eq 1 ]]
then
	fich=$(pwd)"/"$1
	if [[ -f $fich ]]
	then
		read -p "El fichero ya existe. ¿Quiere sobreescribirlo?  y/n " respuesta
		if [[ $respuesta = "y" ]]
		then
			echo "Se piden los datos de los alumnos"
			while [ $respuesta = "y" ]
			do
				pedirDatos
				echo nombre apellidos ": " nota >> $1
				seguimos
				if [[ $respuesta != "y" && $respuesta != "n" ]]
				then
					echo "Debe elegir entre y o n"
					seguimos
				fi
			done
		else
			if [[ $respuesta = "n" ]]
			then
				echo "El programa se detiene"
				exit
			else
				echo "Debe indicar y o n. Vuelva a iniciar el proceso"
			fi
		fi
	else
		echo "El fichero no existe, lo creamos. Escribe los datos de los alumnos"
		respuesta="y"
		while [ $respuesta = "y" ]
		do
			pedirDatos
			echo $nombre $apellidos ": " $nota >> $1
			seguimos
			if [[ $respuesta != "y" && $respuesta != "n" ]]
				then
					echo "Debe elegir entre y o n "
					seguimos
			fi
		done
	fi
else
	echo "Falta el nombre del fichero, vuelva a ejecutar el programa"
fi
```

SOLUCIÓN SEGUNDO SCRIPT

```
#!/bin/bash
# Se comprueba que solo llega un parámetro, que es un fichero y que no está vacío
if [[ $# -eq 1 ]]
then
# Ordeno el fichero por notas
        sort -t":" -k2n $1 > fichOrd.txt
# Construyo la ruta del fichero
        fich=$(pwd)"/fichOrd.txt"

        if [[ -f $fich && -s $fich ]]
        then
# Se lee el fichero línea a línea

                echo "Total de alumnos: " $(wc -l < $fich)
                echo "----------------------------------------"
                echo "| Nombre            | Nota          |"
                echo "----------------------------------------"

                awk -F ':' '{printf "|%-20s|%-17s|\n", $1, $2 }
                {print "----------------------------------------"}' $fich

                echo "Total de alumnos: " $(wc -l < $fich)
                echo "-------------------------------------------------------"
                echo "| Calificación      | Número          |   Porcentaje  |"
                echo "-------------------------------------------------------"
```

En vez de abordar el problema de una manera más tradicional se # plantea esta solución usando awk. Se han mantenido los “echo” # con el objeto de indicar una solución mixta, pero podría hacerse # todo con awk. Como propuesta se puede intentar resumir todas # estas líneas con una sola en un bucle y cambiar los “echo” por # “printf” resolviéndolo totalmente con awk.

```
                awk -F ':' ' $2 == 1 { sum1 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "1", sum1, (sum1*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 2 { sum2 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "2", sum2, (sum2*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 3 { sum3 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "3", sum3, (sum3*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 4 { sum4 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "4", sum4, (sum4*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 5 { sum5 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "5", sum5, (sum5*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 6 { sum6 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "6", sum6, (sum6*100)/NR }' $fich
                echo "-------------------------------------------------------"
```

```
                awk -F ':' ' $2 == 7 { sum7 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "7",
sum7, (sum7*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 8 { sum8 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "8",
sum8, (sum8*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 9 { sum9 += 1}  END {printf "|%-20s|%-17s|%-15s|\n", "9",
sum9, (sum9*100)/NR }' $fich
                echo "-------------------------------------------------------"
                awk -F ':' ' $2 == 10 { sum10 += 1}  END {printf "|%-20s|%-17s|%-15s|\n",
"10", sum10, (sum10*100)/NR }' $fich
                echo "-------------------------------------------------------"
        else
                echo "El fichero debe existir y tener contenido"
        fi
else
        echo "Debe pasarse un solo parámetro"
fi
```

- Para consultas ver:

➢ Página que aclara el uso de los condicionales en bash script:

https://www.atareao.es/tutorial/scripts-en-bash/condicionales-en-bash/#

➢ Sobre awk se puede encontrar mucha documentación en internet, estas dos páginas son un buen ejemplo.

http://www.sromero.org/wiki/linux/aplicaciones/uso_de_awk

https://www.gnu.org/software/gawk/manual/gawk.html