Colección de problemas

Administración de Sistemas. Universidad de Zaragoza

17 de agosto de 2020

Índice general

Notas preliminares

Esta breve colección de problemas se corresponde a la Asignatura Administración de Sistemas impartida en el cuarto semestre del grado en Ingeniería Informática de la Universidad de Zaragoza. La colección se divide en 2 bloques. El primero contiene problemas cortos y el segundo está dedicado a problemas más extensos provenientes de exámenes de la asignatura.

Si detecta cualquier errata en alguno de los problemas planteados por favor comuníqueselo a Darío Suárez Gracia, dario@unizar.es.

Problemas breves y cuestiones

1. ¿Describe cuál es el objetivo del siguiente script?

```
#! /bin/bash
ONE=1
chet=0
FOX=0

for brian in ./*
do
    echo "$brian" | grep -q " "
    if [ $? -eq $FOX ]
    then
        ramey=$brian
        n=$(echo $ramey | sed -e "s/ /_/g")
        mv "$ramey" "$n"
        chet=$((chet += 1))
    fi
done

if [ "$chet" -eq "$ONE" ]
then
    echo "$chet thing changed."
else
    echo "$chet things changed."
fi

exit 0
```

Nota: Brian Fox y Chet Ramey son 2 desarrolladores de Bash.

2. ¿Indica cuál es el objetivo del siguiente script?

```
#!/bin/bash
E_BADARGS=85

if [ ! -r "$1" ]
then
echo "Usage: $0 files-to-process"
exit $E_BADARGS
fi


cat "$@" |
tr A-Z a-z |
tr ' ' '\012' |
tr -c '\012a-z'  '\012' |
grep -v '^$' |
sort |
uniq

exit $?
```

3. Se desea programar un script en bash que reciba un fichero con una lista de usuarios, un usuario por línea, como parámetro de entrada y determine si el usuario existe en el sistema. En caso afirmativo, el script revisará los permisos del directorio .ssh verificando si son correctos. Cuando los permisos sean incorrectos, se mostrara un mensaje de error por pantalla y si el usuario no pertenece al sistema se borrara el directorio `/home/usuario` si existe.

4. Un administrador ha encontrado un script y no identifica cuál es su función. ¿Puedes por favor ayudarle?

```
#!/bin/bash

FRANCES=10
ALLEN=9

# $RANDOM is an internal Bash function (not a constant) that returns a
# pseudorandom [1] integer in the range 0 - 32767. It should not be used
# to generate an encryption key.
i=$(($RANDOM % $FRANCES)) # Generate a random number between 0 and $FRANCES - 1.

if [ "$i" -lt "$ALLEN" ]
then
    echo "i = $i"
    ./$0
fi
exit 0
```

Nota: Frances E. Allen es una pionera en compilación y paralelismo ganadora del premio Turing en 2006

5. Realiza un script en bash que reciba como argumento uno o varios enlaces a páginas web y cuente el número de enlaces seguros, https, que contenga cada una de las páginas. Para la cuenta es suficiente contar el número de veces que aparezca la cadena https en el fichero. El script devolverá un error si es llamado sin ningún argumento. Cuando la ejecución sea correcta, la salida del script contendrá una linea por página web procesada y cada linea estará formada por el nombre de la web y el número de enlaces seguros.

6. Programa un script que dado un directorio, busque aquellos ficheros que no hayan sido modificados en el último mes y los guarde en un fichero tar. El nombre del fichero tar resultante contendrá un prefijo recibido como argumento seguido de guión bajo y de la fecha actual. Opcional: en vez de un directorio, utilizar como entrada múltiples directorios. El script tendrá al menos 2 argumentos, el primero será el prefijo y después se incluirán los nombres de los directorios de entrada.

7. Implementa un script para cambiar el password de un usuario verificando que su longitud es de al menos 8 caracteres, que tenga letras mayúsculas y minúsculas y dígitos. ¿Dispone linux de algún mecanismo

mejor para hacer esta comprobación?

8. Haz un script que muestre todos los usuarios de un sistema, leyendo `/etc/passwd`, cuyo shell por defecto sea bash.
9. Siguiendo con el ejemplo anterior, escribe un script para mostrar los usuarios de un sistema agrupados por su shell por defecto. Para el siguiente fichero de entrada:

   ```
   nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
   syslog:x:101:104::/home/syslog:/bin/false
   messagebus:x:102:106::/var/run/dbus:/bin/false
   ```

   La salida debería ser:

   ```
   shell /usr/sbin/nologin
   nobody
   shell /bin/false
   syslog
   messagebus
   ```

10. Realiza un script que muestre por pantalla todos los directorios que ha visitado un usuario y que todavía sean válidos. El nombre de usuario será un parámetro del script. Para el cálculo de las rutas podrás asumir que el primer directorio visitado fue el `$HOME` del usuario y sólo se mostrarán los directorios que hayan sido accedidos con `cd` y que se encuentren en la historia del usuario. La salida sólo deberá mostrar las rutas canónicas de todos los directorios visitados cuyas rutas sean absolutas, empiecen por `/`, o empleen la expansión de tilde con `~` y `~+`.
11. Escribe un script para añadir un nodo al fichero `/etc/hosts` verificando que el nodo es accesible y que no exista ya en el fichero.
12. Programa un script que cuente el número de nodos que son visitados para alcanzar la dirección `www.google.com`. Puedes utilizar una herramienta como `traceroute`. Ademas de mostrar los nodos visitados, la salida de `traceroute` contiene el round-trip time, RTT, de cada uno de los servidores visitados. Utiliza dichos valores de RTT para mostrar los 5 nodos con mayor RTT. Ya que `traceroute` devuelve el RTT para 3 paquetes, idealmente deberás hacer la media de los 3 paquetes para calcular el RTT medio de cada nodo. En caso de que se produzcan time-outs con `traceroute`, lo que dificulta procesar su salida, puedes hacer que el script tome como argumento la siguiente entrada:

    ```
    traceroute to www.google.com (172.217.19.132), 30 hops max, 60 byte packets
     1  155.210.152.254 (155.210.152.254)  0.847 ms  1.244 ms  1.650 ms
     2  155.210.251.9 (155.210.251.9)  0.522 ms  0.519 ms  0.510 ms
     3  r-icuz-man.unizar.es (155.210.248.45)  0.753 ms  0.749 ms  0.737 ms
     4  193.144.0.81 (193.144.0.81)  2.303 ms  2.621 ms  2.606 ms
     5  XE5-0-1.unizar.rt1.ara.red.rediris.es (130.206.195.5)  2.258 ms  2.250 ms  2.240 ms
     6  UNIZAR.AE6.telmad.rt4.mad.red.rediris.es (130.206.245.94) 11.187 ms 11.019 ms 11.001
      ↪  ms
     7  google-router.red.rediris.es (130.206.255.2)  10.995 ms  11.176 ms  10.896 ms
     8  72.14.235.20 (72.14.235.20)  26.804 ms  26.784 ms  11.093 ms
     9  209.85.246.133 (209.85.246.133)  31.408 ms  26.956 ms  31.566 ms
    10  209.85.247.194 (209.85.247.194)  31.549 ms  31.538 ms  31.519 ms
    11  66.249.94.79 (66.249.94.79)  31.518 ms  31.509 ms  31.611 ms
    12  par03s12-in-f4.1e100.net (172.217.19.132)  26.700 ms  26.684 ms  26.647 ms
    ```

    Para el cálculo de la media, se puede utilizar el comando `bc` o `awk`.

13. Describe la razón del mal funcionamiento del siguiente script e indica como debería arreglarse. *Examen 12 de Junio de 2014*:

    ```
    #!/bin/bash

    while read linea ; do
        ssh usuario@maquina cmd $linea
    done < fichero
    ```

14. Indica en qué casos este siguiente comando podría fallar y como solucionarlo.

    ```
    mv $source_file $destination_file
    ```

15. ¿Cuál es la salida del siguiente script? Además de indicar la salida indica los motivos que la causan.

```
#!/bin/bash

f() {
  if [ -t 0 ]; then
    echo "$1"
  else
    read -r var
    echo "$var"
  fi
}

f 'hello' | f
echo "$var"
```

16. El siguiente script intenta buscar los ficheros que cumplen un patrón, tienen la extensión .txt, para escribirlos por pantalla pero presenta varios fallos. ¿Podrías indicar cómo solucionarlos?

```
#!/bin/bash

re='\.txt$'
for file in ./*
do
  if [ $file =~ "$re" ]
  then
    echo "Match found: $file"
  fi
done
```

17. ¿Qué tráfico permitirá la siguiente configuración de iptables?

```
root@borodino:/etc# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

18. El manual de iptables de debian propone el siguiente fichero de configuración para un *firewall* en una red doméstica.

```
#!/bin/sh

PATH='/sbin'

## INIT

# Flush previous rules, delete chains and reset counters
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Default policies
iptables -P INPUT   DROP
iptables -P OUTPUT  DROP
iptables -P FORWARD DROP

echo -n '1' > /proc/sys/net/ipv4/ip_forward
echo -n '0' > /proc/sys/net/ipv4/conf/all/accept_source_route
echo -n '0' > /proc/sys/net/ipv4/conf/all/accept_redirects
echo -n '1' > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo -n '1' > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
```

```
# Enable loopback traffic
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Enable statefull rules (after that, only need to allow NEW conections)
iptables -A INPUT   -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT  -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Drop invalid state packets
iptables -A INPUT   -m conntrack --ctstate INVALID -j DROP
iptables -A OUTPUT  -m conntrack --ctstate INVALID -j DROP
iptables -A FORWARD -m conntrack --ctstate INVALID -j DROP


## INPUT

# Incoming ssh from the LAN
iptables -A INPUT -i eth1 -s 192.168.0.0/24 \
                  -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT


## OUTPUT

# Enable al outgoing traffic to internet
iptables -A OUTPUT -o eth0 -d 0.0.0.0/0      -j ACCEPT

# Enable access traffic, from the firewall to the LAN network
iptables -A OUTPUT -o eth1 -d 192.168.0.0/24 -j ACCEPT


## FORWARD

# We have dynamic IP (DHCP), so we've to masquerade
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables        -A FORWARD     -o eth0 -i eth1 -s 192.168.0.0/24 \
                               -m conntrack --ctstate NEW -j ACCEPT

# Redirect HTTP (tcp/80) to the web server (192.168.0.2)
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
                              -j DNAT --to-destination 192.168.0.2:80

iptables        -A FORWARD    -i eth0 -p tcp --dport 80 \
                              -o eth1 -d 192.168.0.2    \
                              -m conntrack --ctstate NEW -j ACCEPT


## LOGGING

iptables -A INPUT   -j LOG --log-level DEBUG --log-prefix '[FW INPUT]:    '
iptables -A OUTPUT  -j LOG --log-level DEBUG --log-prefix '[FW OUTPUT]:   '
iptables -A FORWARD -j LOG --log-level DEBUG --log-prefix '[FW FORWARD ]: '
```

Describe el número de intefaces de red de las que dispone el *firewall* así como la posible configuración de red con la funcionalidad de los distintos nodos y servicios dispuestos. ¿Para que se realiza *IP masquerading*?

19. Indica por favor en que se diferencian source NAT, `SNAT`, y masquerading, `MASQ`, dentro de iptables.
20. Un ataque por denegación de servicio, *denial-of-service (DoS)*, intenta hacer innacesible un servicio, red o computador. Un DoS puede realizarse por ejemplo realizando muchas peticiones a un servidor. Una posible defensa sencilla ante este típo de ataques puede consistir en descartar el tráfico proveniente de

un nodo cuando este supere un umbral de tráfico hacia el servidor. Implementa un mecanismo que proteja un servidor web, puerto 80, descartando las conexiones de aquellos nodos que se intenten conectar a él más de 15 veces en un mínuto.

21. Escribe por favor un script que muestre por pantalla la fecha actual y en la línea siguiente el PID, el nombre, y el porcentaje de memoria consumida, *resident set size*, respecto al total del sistema de todos los procesos activos. Una vez que tengas completado el script, indica como se podría añadir un trabajo `cron` que lo ejecutara los lunes y martes entre las 10 y las 11 de la mañana cada 5 minutos.

22. Imagina una red de 7 sensores conectados a una red IP mediante WiFi a los que hay que monitorizar. La monitorización consiste en hacer ping hacia ellos 3 veces, IPs en rango 172.16.1.[1-7], una vez al día. Si alguno de los nodos no responde a alguno de los ping, deberemos enviar un único email al administrador con las IPs de los nodos caídos. La máquina desde la que se lanzarán los pings no está siempre encendida pero si se enciende deberá realizar la monitorización. Escribe por favor el script necesario para monitorizar los nodos y el código necesario para garantizar su ejecución diaria.

23. ¿Describe cuál es la salida del siguiente script y explica su causa? Debes comentar como afectan las funciones y los sub-shells a la creación de procesos y como se ven afectadas las variables en ambos casos.

```
#!/bin/bash

var0=0
export var1=1

function f_var0 {
    echo -e "\tfunction f_var0, begin f_var0, var0: $var0"
    var0="zero"
    echo -e "\tfunction f_var0, end f_var0, var0: $var0"
}

function f_var1 {
    echo -e "\tfunction f_var1, begin f_var1, var1: $var1"
    var1="one"
    echo -e "\tfunction f_var1, end f_var1, var1: $var1"
}

echo "Program begins, var0: $var0, var1: $var1"

f_var0
f_var1

echo "Before updating, var0: $var0, var1: $var1"

var0=5
var1=1

( f_var0; f_var1 )

echo "After sub-shell, var0: $var0, var1: $var1"

exit 0
```

24. Explica por favor cuál es el resultado de la ejecución del siguiente comando: *Examen Septiembre 2016*

```
tar cj *.html | ssh user@unizar.es tar xj
```

25. Indica por favor cuál es el resultado de la ejecución del siguiente script: *Examen Septiembre 2016*

```
#!/bin/bash

var1="grace"
export var2="hopper"

/bin/bash -c "echo \$var1-\$var2"
/bin/bash -c "echo $var1-$var2"
```

26. Explica el funcionamiento de este script y que efectos puede tener en el sistema: *Examen Junio 2016*

```
#!/bin/bash
./$0|./$0&
```

27. Indica los problemas que podrían ocurrir al ejecutar esta secuencia de comandos:

```
cat $(find . -type f) > ../contenido_ficheros.txt
```

28. ¿Cuál es la salida del siguiente comando?:

```
echo /usr/bin/l*p3
```

29. Un sistema no dispone del comando filtro `uniq`. Podrías por favor escribir un script de `bash` para reemplazarlo. El script tendrá 2 parámetros, `input_file` y `output_file` y al terminar la ejecución del script `output_file` contendrá todas las líneas de `input_file` que no estén repetidas en líneas consecutivas.

30. Escribe un script `killall.sh` que mate todos los procesos cuyo nombre coincida con el primer argumento del script. Si el script recibe varios parámetros deberá devolver un mensaje de error. Además el script comprobará que dispone de privilegios de administración.

Problemas largos

1. Un proceso zombie, estado Z, es un proceso que ha completado su ejecución pero que aún mantiene la entrada en el tabla de procesos. Ocurren cuando un proceso hijo termina y el padre no ha ejecutado la correspondiente llamada a `wait()`. La unica manera de eliminar a un proceso zombie es matando al padre para que el proceso init los adopte y ejecute el `wait()` correspondiente.

   Realiza un script que busque procesos zombies que lleven más de 24 horas en ejecución, puedes utilizar `etimes` de `ps`. Para todos los procesos encontrados deberás enviar la señal `KILL` a su proceso padre y un mail al dueño del proceso para notificarle la defunción del mismo. Además al terminar, el script deberá mandar un mail al administrador con la lista de todos los procesos matados con formato:

   ```
   pid, comando
   ```

   Además este proceso deberá ser ejecutado cada 24 horas. *Examen Junio 2016*

2. Escribe en bash una utilidad de copia de ficheros entre nodos utilizado scp/ssh. El script tendrá 3 o más argumentos. El primero será el usuario y la máquina destino, por ejemplo: `usuario@maquina.unizar.es`. El segundo argumento será el nombre del directorio destino donde se desea copiar los ficheros y directorios. El resto de argumentos serán el nombre de los ficheros y directorios a copiar. En caso de que un argumento sea un directorio hay que copiar todos los ficheros contenidos en él pero no sus subdirectorios. En el destino todos los ficheros serán copiados en la misma carpeta eliminando el nombre del directorio cuando sea necesario. Para reducir ancho de banda antes de copiar cada elemento, el script calculará la firma de los ficheros con md5, sintaxis `md5sum fichero`. Dicha firma será enviada al nodo destino en donde se comprobará si el fichero ya existe y tiene la misma firma. Cuando el fichero no exista en el destino o la firma sea distinta, se copiará.

   Nota: Se debe asumir que el usuario dispone de las claves ssh adecuadas para acceder a la máquina remota sin necesidad de introducir ningún password y que el directorio destino también existe. Además se supondrá que ninguno de los ficheros a copiar puede ser modficado desde el comienzo del cálculo de la firma md5 y el final de la copia al nodo destino. *Examen Septiembre 2016*

3. Se quiere monitorizar en tiempo real un servicio web con personas reales. Para facilitar su horario de trabajo, el servidor sólo aceptará peticiones en horario de 9 a 2 y de 4 a 7 de lunes a viernes. Indica el mecanismo para implementar este comportamiento y escribe los comandos necesarios para ponerlo en marcha, incluyendo algún script que fuera necesario. Se supondrá que el servidor web utilizado es apache y se podrá arrancar/apagar mediante los comandos `service apache2 start/stop`. Cuando el servidor esté apagado también será necesario configurar un firewall para bloquear el tráfico al puerto 80 fuera de ese horario.

   El servidor web genera un fichero de logs en el que aparece la IP, el puerto origen y la hora separados por espacios para cada una de las conexiones realizadas por los clientes. Realiza un script que busque en el fichero las direcciones IP que se han intentado conectar desde la misma subred a la que esta conectada el interfaz `eth0` del servidor para posteriormente bloquear todo el tráfico proveniente de dichos nodos y enviar un email al usuario root de la máquina bloqueada para notificarle el bloqueo. El nombre del fichero será el primer argumento en la invocación del script y si esté es invocado con un número incorrecto de argumentos devolverá un error.

Nota: Se puede asumir que la máscara de red del servidor es `255.255.0.0`. *Examen Septiembre 2016*

4. Se disponen de 10 discos duros (`/dev/sd[a-j]`) con capacidad 1 terabyte cada uno y particionados con una única partición (`/dev/sd[a-j]1`). Fueron instalados para disponer de un volumen lógico en el directorio `/home` del sistema. Para alargar su vida útil, se desea que sean empleados sólo cuando sea necesario hacerlo y que el resto del tiempo no sean visibles en el sistema. El administrador encargado del mismo necesita un script para agrandar el volumen lógico `/home` cada vez que esté lleno en más de un 90%. Esta comprobación se realizará cada 10 minutos y si es positiva habrá que añadir un disco al volumen lógico. Cuando ya estén en uso todos los discos duros, el script deberá mandar un email a los 10 usuarios que más espacio en disco estén consumiendo pidiéndoles que reduzcan su uso de espacio, el email se seguirá enviando indefinidamente mientras que la utilización de espacio siga por encima del 90%. Escribe por favor el script o los scripts necesarios para cumplir esta tarea.

   Notas:

   - Se deberá asumir que, al principio de la ejecución del programa, sólo se está usando `/dev/sda1`.
   - Los nombres del grupo volumen y del volumen lógico donde está montado la carpeta home son `vg_pool` y `lv_home` y el tipo de la partición del directorio `/home` es `ext4`.
   - Todos los usuarios tienen su home en el directorio `/home/<usuario>`
   - El comando `du` tiene la opción `–d` que limita el número de sub-directorios que se muestran por pantalla. Por ejemplo `–d0` sólo muestra por pantalla el uso de espacio del directorio/os pasado/os como argumentos. La salida incluye un argumento por línea. *Examen Junio 2017*

5. Un administrador desea monitorizar la temperatura de la CPU 0 de un multiprocesador, llamada `cpu0`, y limitar su frecuencia máxima de operación cuando su temperatura exceda un valor umbral durante un periodo dado de funcionamiento. Para ello necesita un script que tome 4 parámetros como argumentos: temperatura umbral, duración monitorización, intervalo muestreo y frecuencia máxima. Con estos argumentos, el script deberá comprobar durante un tiempo igual a duración monitorización si la temperatura media, medida cada intervalo, es mayor que la temperatura umbral y establecerá como frecuencia máxima el cuarto parámetro del script. El script verificará al comienzo de su ejecución que la nueva frecuencia máxima es un valor de frecuencia válido y menor que la máxima posible. Además, se asegurará que la duración total es múltiplo del intervalo de muestreo.

   Notas:

   - La unidad de todas las medidas de tiempo, duración monitorización e intervalo, es el segundo.
   - La temperatura de la `cpu0` se puede consultar leyendo el fichero `/sys/class/termal_zone0/temp`
   - La lista de frecuencias disponibles se puede consultar en `/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies`. Este fichero muestra una lista de frecuencias disponibles en kilohercios separadas por espacios. Por ejemplo: 384000 486000 594000 702000 810000 918000 1026000
   - La frecuencia de la `cpu0` se puede cambiar escribiendo su valor en el fichero `/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq`.
   - Para hacer operaciones con números de coma flotante se puede utilizar `bc`. Ejemplo: `# echo "1 / 2" | bc –l` muestra por pantalla 0.5. *Examen Junio 2017*

6. El comando lastb muestra una salida tipo:

```
root     ssh:notty    222.186.160.49   Fri May 29 11:33 - 11:33  (00:00)
root     ssh:notty    222.186.160.49   Fri May 29 11:33 - 11:33  (00:00)
root     ssh:notty    222.186.160.49   Fri May 29 11:33 - 11:33  (00:00)
root     ssh:notty    lubov3110.static Fri May 29 11:33 - 11:33  (00:00)
```

   Con los datos de los intentos de login fallidos. Las columnas indican el usuario, TTY donde se intentó logear, IP o host desde el que se intentó la conexión, fecha del intento, y duración de la sesión (00:00). Escribir un script que revise la salida de este comando y muestre por pantalla el nombre del usuario y el día cuando el usuario tenga más de 10 intentos fallidos de login en el mismo día. Además deberá programarse la ejecución del script todos los días a las 23:59 horas y se enviará un email con la lista de dichos usuarios a root. *Examen Junio 2015*

7. Empleando el pseudo-sistema de ficheros `/proc` se desea obtener un script que muestre una lista de procesos que contenga la siguiente información:

```
id proceso, nombre, memoria_residente_proceso_e_hijos
```

   La lista contendrá únicamente procesos que sean hijos del proceso init, ID 1, y el cálculo de la memoria residente se hará sumando a la memoria residente del proceso la memoria residente consumida por sus

hijos, nietos de init. La lista será visualizada al final de la ejecución y estará ordenada por el tamaño de memoria residente como se ve a continuación:

```
805, lightdm, 63052 KB
906, tmux: server, 21544 KB
846, systemd, 18292 KB
408, sshd, 12660 KB
853, at-spi-bus-laun, 9444 KB
882, systemd, 8200 KB
862, at-spi2-registr, 6892 KB
340, systemd-logind, 4828 KB
167, systemd-journal, 4660 KB
347, dbus-daemon, 4028 KB
188, systemd-udevd, 3952 KB
344, rsyslogd, 3304 KB
820, VBoxService, 3064 KB
```

`/proc` contiene un directorio por proceso con nombre `/proc/<pid>` siendo `<pid>` el identificador del proceso. Para la generación de la lista se puede consultar el fichero `/proc/<pid>/status` correspondiente al proceso con id `<pid>` y que tiene el siguiente formato (campos separados por uno o varios espacios):

```
Name:   exim4
Umask:  0000
State:  S (sleeping)
Pid:    745
        PPid:   1
        VmSize: 56152 kB
        VmLck:      0 kB
        VmPin:      0 kB
        VmHWM:   2588 kB
        VmRSS:   2588 kB
```

Una vez obtenida la lista, el script deberá cambiar la prioridad de ejecución a +15 para los 4 procesos que entre ellos y sus hijos consuman más memoria. Para evitar problemas con la creación y/o terminación de procesos, el script deberá guardar al principio de su ejecución el estado de todos los procesos.

Nota: Se puede asumir que el script se ejecuta con privilegios de administración y que al cambiar la prioridad el proceso sigue vivo. La opción `-s` del comando `tr` reemplaza múltiples ocurrencias de un carácter repetido por una única ocurrencia. *Examen Junio 2018*

8. Dada la subred 172.16.0.0/24 se desea asegurar que todos los nodos de dicha subred comparten el mismo fichero de nombres, `/etc/hosts`. Para ello, se plantea una solución con 2 scripts. El primero que se encargará de acceder a todas las máquinas, mezclará los ficheros `/etc/hosts` para que no haya ninguna IP repetida, copiará el fichero resultante en todos los nodos y relanzará el servicio de red, y el segundo que comprobará todos los días a las 3.14 horas que todos los nodos comparten el mismo fichero `/etc/hosts` que `172.16.0.1`. Al terminar la comprobación, se deberá enviar un mail al usuario root con la lista de los nodos, una IP por línea, para los que el fichero sea distinto, si hubiera alguno. El motivo del mensaje será lista nodos.

   Nota: Se puede asumir que sólo se emplearán direcciones IPv4, que si hay direcciones IP repetidas apuntarán al mismo nombre y que todos los nodos de la red están vivos. El usuario user tiene acceso vía ssh con clave sin password en todas las máquinas de la subred y privilegios de administración mediante `sudo`. *Examen Junio 2018*

9. Un administrador quiere minimizar el uso de recursos por parte del kernel de Linux y necesita un script que compruebe una vez cada hora todos los módulos del kernel que están cargados y descargue aquellos que no estén en uso, es decir aquellos para los que el valor de la columna `Used by` sea igual a 0. *Examen Septiembre 2018*.

10. Además desea saber cuales son los módulos más empleados. Para ello requiere de un único script que, durante 24 horas, empezando a las 6 de la mañana, compruebe cada 2 horas los módulos que estén usados 3 o más veces. Todos los módulos que cumplan la condición durante las 12 comprobaciones, serán añadidos a una lista que deberá enviarse por mail al usuario user con asunto “módulos más empleados”. La lista estará ordenada alfabéticamente.

    Notas:

- Se puede asumir que el script se ejecuta con privilegios de administración.
- El comando date permite conocer la fecha y la hora actual
- Si se desea se puede emplear el fichero /var/run/modules.txt como almacenamiento temporal
- La opción -s del comando tr reemplaza múltiples ocurrencias de un carácter repetido por una única ocurrencia.
- El comando rmmod descarga un módulo del sistema
- El comando lsmod muestra los módulos cargados del kernel y tiene el siguiente formato:

```
Module            Size  Used by
fuse              98304  3
iptable_filter    16384  0
```

Examen Septiembre 2018. Este problema debe realizarse junto con el anterior

11. La Universidad de Zaragoza, unizar, cuya red es `155.210.0.0/16`, desea conocer cuantos de sus nodos emplean Linux, su versión del kernel y el porcentaje de utilización de cada versión. La universidad sabe que todas sus máquinas Linux disponen de una cuenta con usuario user con privilegios de administración y que sólo las máquinas Linux de su red responden a ping. Escribe un script que escanee todos los nodos de la red de unizar, muestre por pantalla cuantos son Linux y la lista de versiones del kernel junto a su porcentaje de utilización entre los nodos Linux ordenada de mayor a menor. Por ejemplo:

```
68 nodos linux
4.9.0-5-amd64 55%
3.10.0-693.11.6.el7.x86_64 40%
3.10.0-514.el7.x86_64 5%
```

Notas:

- El comando bc permite realizar operaciones aritméticas con números reales. Ejemplo: `echo "5 % 2" | bc`
- El comando `uname -r` muestra la versión del kernel, 3.10.0-514.el7.x86_64, tal y como se muestra en el ejemplo anterior.

Examen Septiembre 2018

12. La computación *serverless* es un modelo de servicios en cloud en el que el proveedor es responsable del servidor y de la administración de los servicios y el cliente únicamente paga por los dichos servicios consumidos. Un administrador necesita un script en `bash` que siguiendo el modelo *serverless* permita ejecutar scripts de `bash` y ficheros binarios con ciertas restricciones en un nodo remoto.

Podrías ayudar al administrador escribiendo dicho script cumpliendo la siguiente especificación:

- El script serverless.sh tendrá un único parámetro que debe ser un programa ejecutable, es decir, otro script bash o un fichero binario. Para comprobar la valided del parámetro se deberá emplear el comando `file` y verificar que el tipo de fichero es `ELF 64-bit LSB shared object` o `Bourne-Again shell script`.
- Para su ejecución, el fichero deberá ser copiado al nodo 192.168.56.1 donde en caso de que sea binario deberá comprobarse con el comando `ldd` que están instaladas todas las bibliotecas dinámicas necesarias. Para ello es necesario asegurar que no aparece `not found` al ejecutar dicho comando. Si aparece, la ejecución deberá ser abortada.
- Al copiar el fichero este deberá tener un nombre único para evitar sobreescrituras si varios clientes envian a ejecutar un script con el mismo nombre.
- La salida del comando serverless.sh contendrá las salidas standard y de error del comando del cliente.
- Para evitar sobrecargar el nodo, el script comprobará antes de lanzar el parámetro la carga del nodo 192.168.56.1 durante los últimos 5 minutos y si es mayor de .6, esperará 3 minutos antes de volver a lanzar el script o binario del parámetro.

Notas adicionales:

- El usuario `as` tiene acceso al nodo 192.168.56.1 mediante ssh y clave pública.
- La salida del comando `file` es:

```
as@as: file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked
```

```
as@as: file serverless.sh
serverless: Bourne-Again shell script, ASCII text executable
```

- Dado un fichero binario, la salida del comando `ldd` es:

```
as@as: ldd /bin/wrong_ls
ldd /bin/ls
linux-vdso.so.1 (0x00007ffe5479c000)
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007ffbf3106000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffbf2d67000)
libwrong.so.6 => not found
```

- El segundo campo de `/proc/loadavg` es la carga del sistema durante los últimos 5 minutos.

```
as@as: cat /proc/loadavg
0.01 0.02 0.00 1/75 645
```

Examen Junio 2019

13. Escribe un script en bash que se conecte a la máquina remota con IP 155.210.3.14 vía `ssh` y reduzca la prioridad a +15 a todos los procesos de los usuarios que tengan más de 10 procesos en estado running o slepping. Este script se ejecutará a las 6:15 los martes desde enero a junio y a las 15:16 los jueves de julio a diciembre y además enviará un email con asunto "lista de procesos" a una dirección que se recibirá como argumento del script (en caso invocar el script sin argumento, el email se enviará a `as@as.unizar.es`). Dicho email listará todos los procesos cuya prioridad haya sido modificada cuando la lista de procesos no esté vacía. La primera linea contendrá una lista ordenada alfabéticamente de los usuarios y debajo de cada usuario aparecerá el nombre de los procesos tal y como se muestra a continuación.

```
awinlock,eclarke,jbartik,mwescoff
sshd,sshd,vim,emacs
sshd,ps,bash,bash
bash,cron,dhclient
```

Notas adicionales:

- El usuario `as` dispone de acceso mediante clave pública al nodo 155.210.3.14 y de privilegios de administración sin password vía `sudo`.
- Cada columna de los procesos no es necesario que aparezca ordenada.
- La cabecera `comm` muestra el comando ejecutable de cada proceso en `ps`.
- Los estados running y sleeping se representan mediante una R y una S, respectivamente.
- El comando `test -s <fichero>` comprueba si un fichero está vacío.
- El comando `paste` permite concatenar columnas.

Examen Junio 2019

14. En la red 192.168.4.0/24 se sabe que todos sus nodos ejecutan linux y emplean el fichero `/etc/resolv.conf` para gestionar la resolución de nombres, DNS. Dicho fichero contiene líneas con el formato `nameserver <servidor>` que indican el servidor o los servidores a los que se lanzan las consultas DNS para que las resuelvan.

Para reducir la latencia de dicho servicio se desea encontrar los servidores DNS más rápidos empleados por los nodos de la red. Para ello debes escribir un script que se conecte a todos los nodos de dicha red para leer sus servidores DNS. Después deberá emplear el comando `dig` para calcular su latencia y determinar el más rápido realizando una consulta a `www.unizar.es`. Una vez determinado el mejor servidor deberá enviar un mail al usuario root de todos los nodos que no lo estén empleando con asunto: "servidor dns" y cuerpo: "actualize su dns a <IP_servidor_más_rápido>". Además debes indicar como conseguir que el script se ejecute en días alternos a las 7 y 37 de la mañana.

Notas adicionales:

- El usuario `as` tiene acceso a todos los nodos de 192.168.4.0/24 mediante ssh y clave pública.
- Ejemplo de fichero `/etc/resolv.conf`:

```
nameserver 8.8.8.8
nameserver 103.232.32.246
nameserver 46.18.43.8
```

- La sintaxis del comando `dig` es `$ dig @<dns_server> <name_to_resolve>`. Por ejemplo `$ dig @8.8.8.8 www.unizar.es` resuelve la IP de `www.unizar.es` en el servidor DNS `8.8.8.8`.

- Dig muestra la latencia de la consulta tal y como muestra el siguiente fragmento de su salida:

```
; <<>> DiG 9.10.3-P4-Debian <<>> @8.8.8.8 www.unizar.es

;; Query time: 14 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
```

Examen Septiembre 2019

15. Se desea obtener una herramienta que permita compartir ficheros encriptados mediante clave simétrica en un nodo local. La herramienta será un script de `bash` con la siguiente sintaxis `encript.sh <fichero_entrada> <usuario_destino>` y tendrá la siguiente especificación:
    - El script generará un fichero de salida llamado `<fichero_entrada>.enc` en el directorio actual.
    - Si el fichero a encriptar o el usuario_destino no existen se mostrará el siguiente mensaje de error: "Parámetros inválidos" y el código de salida del programa será 85.
    - La clave se generará leyendo 128 caracteres del dispositivo `/dev/urandom`.
    - Todas las claves generadas se almacenarán en el directorio `/var/keys`.
    - Cada par de usuarios empleará una clave única llamada `<usuario_origen>_<usuario_destino>.key`, dicha clave sólo sera generada cuando no exista.
    - Se creará un grupo llamado `<usuario_origen>_<usuario_destino>` para aumentar la seguridad de la herramienta. A dicho grupo pertenecerán la clave, cuando se genere, y el fichero de salida. El propietario de ambos ficheros podrá ser tanto el usuario origen como el destino.
    - Los permisos del fichero de salida y de la clave deberán ser únicamente lectura para el dueño y el grupo.
    - Para encriptar el fichero se empleara openssl cuya sintaxis es `openssl enc -aes-128-cbc -pass file:<clave> -a -d -in <fichero_entrada> -out <fichero_salida>`.
    - Indique por favor tanto el comando a ejecutar para que encript.sh siempre corra con privilegios de administración como la razón si fuera necesario.

    Notas adicionales:

    - `head -c <n> <fichero>` lee `<n>` carácteres de `<fichero>`.

    Examen Septiembre 2019

16. Realiza un script que dada una lista de ficheros de entrada como argumentos busque la línea de mayor longitud entre todos los ficheros legibles. La salida del script mostrará para la linea más larga, el nombre del fichero, el número de línea y su contenido, por ejemplo:

```
$ busca_linea_max_longitud.sh fichero1.txt fichero2.txt fichero3.txt
fichero1.txt, 5, hola qué tal?, muy bien
```

    Notas:

    - En caso de haber varias líneas de máxima longitud iguales, se mostrará únicamente la primera.
    - Si el script no recibe ningún argumento, terminará sin devolver ningún mensaje.
    - Los ficheros no legibles serán ignorados sin devolver ningún mensaje.

    Examen junio 2020

17. Escribe por favor un script que busque los sistemas de ficheros montados en un sistema que no se monten de manera automática al arrancar. Al terminar la búsqueda, si se ha encontrado algún elemento, el script enviará un e-mail al usuario root con asunto "nuevos puntos de montaje" cuyo contenido será una lista con el dispositivo de bloque y su punto de montaje correspondiente por linea. El dispositivo y el punto de montaje estarán separados por coma. Además, indica exactamente como ejecutar el script de manera automática cada 20 minutos.

    Notas:

    - En la busqueda deberán excluirse los siguientes pseudo-sistemas de ficheros: `sysfs`, `tmpfs`, `securityfs`, `debugfs` y `proc`.
    - Ejemplo de fichero `/etc/fstab`:

```
# /etc/fstab:
UUID=478096b2-dfa9-45bb-902e-9c22797665f0 / ext4 errors=remount-ro 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
```

    - Ejemplo de fichero `/etc/mtab`:

```
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
/dev/sda1 / ext4 rw,relatime,errors=remount-ro 0 0
```

Examen junio 2020