```
Problemas shell_1
# Sustituir todas las ocurrencias de “No” por “XX”.
sed 's/No/XX/g' #prueba.txt
# Insertar “>>” al principio de cada línea de un
texto
sed 's/^/>> &' #prueba.txt
# Eliminar los dos últimos caracteres de cada línea.
sed 's/...$//' #prueba.txt
# Eliminar la extensión de un archivo (ejm: foo.txt →
foo).
sed -e 's/[.][[:alnum::]*//' #prueba.txt
# Elimine todas las vocales de un texto.
sed 's/[aeiouAEIOU]/g' #prueba.txt
# Sustituya los espacios seguidos por un sólo espacio
(ejm: foo bar fooz → foo bar fooz)
sed 's/\ \{2,\}/ /g' #prueba.txt
# Elimine los 10 primeros caracteres de cada línea.
sed 's/^.\{0,10\}//g'
# Buscar en /etc/passwd usuarios del sistema con id >
99
awk -F ':' '$3>99' /etc/passwd
# Ordenar los usuarios del apartado anterior por id.
awk -F ':' '$3>99' /etc/passwd | sort -n -t ":" -k3
# Dado un fichero con nombres y otro con apellidos,
unir los dos ficheros y dejar sólo una de
# las personas con apellido repetido, mostrando
cuantas personas de cada familia hay.
paste -d ' ' nombres.txt apellidos.txt | sort -k2 |
uniq -f1 -t ' '
# Eliminar las lineas vacías de un texto.
sed 's/^$//g' #fichero
# Extraer hora y minuto de la fecha: 11:26
date +"%H:%M"
# Formatear la fecha con la forma: abr 11, 2013
date +"%h %d, %Y"
```

```
Problemas shell_2
# Dado un fichero con nombres y apellidos, generar
una dirección de correo @unizar para cada línea, que
contenga la primera letra del nombre y todo el
apellido.
sed -r 's/^(.)[[:alpha:]]+[
]([[:alpha:]]+)/\1\2@unizar.es/' $1 | tr [A-Z] [a-z]
awk '{print tolower(substr($1,0,2) $2 "@unizar.es")
}' $1
# Un profesor tiene las notas de los alumnos que se
presentaron a un examen en un fichero de texto. Cada
línea está compuesta por el nombre y apellido del
alumno, un espacio y su calificación, que siempre
será apto o no apto. Desea que para cada línea del
texto se genere lo siguiente:
# mailto: direccion@unizar.es body: nombre apellido
nota. Presentados: n, aprobados: m
if [ $# -ne 1 ]; then echo -e "\nusage:
./shell_tools_2.1.sh <file>\n"; exit 1; fi
awk ' BEGIN{ap=0}
    {
      mail=tolower(substr($1,0,2) $2 "@unizar.es");
if ($3 >= 5) ap++;
      printf "mailto: %s body: %s %s %d. Presentados:
%d, aprobados: %d\n",mail,$1,$2,$3,NR,ap
    }' $1
# Tenemos un directorio que contiene, entre otras
cosas, scripts de shell. Se desea modificarlos,
insertando entre su primera y segunda línea el
copyright del autor y la fecha:
if [ $# -ne 2 ]; then echo -e "\nusage:
./shell_tools_2.1.sh <files_directory> <licence_file>
\n"; exit 1; fi
if [ ! -d $1 ]; then echo "$1 no es un directorio.";
exit 1; fi
if [ ! -f $2 ]; then echo "$2 no es un fichero
leible."; exit 1; fi
for i in $(ls $1)
do
    if [ -f $i ]; then sed "1a# $(<$2)\n# $(date)\n"
$i; fi
done
 # Hacer un shell script que dado un fichero de texto,
 cuente las cinco palabras más repetidas
 tr -c '[:alnum:]' '[\n*]' < $1 | sort | uniq -c |
 sort -nr | head  -6
 # Partiendo de esta utilidad realice un shell-script
 'mi_banner.sh' que admita hasta tres argumentos de
 tamaño máximo de cuatro caracteres tal que el comando
 './mi_banner.sh hola jose luis'
 log() {
     echo -e "\nusage: ./shell_tools_2.5.sh <word>
 <word> <word>\n"
     echo -e "\t<word>\t\tWord to print as banner. Max
 length: 4.\n"
     exit 1
 }
 if [ $# -ne 3 ]; then echo "error -- too few
 parameters."; log; fi
 if [ ${#1} -gt 4 ]; then echo "error -- incorrect
 first parameter length, max 4."; log; fi
 if [ ${#2} -gt 4 ]; then echo "error -- incorrect
 second parameter length, max 4."; log; fi
 if [ ${#3} -gt 4 ]; then echo "error -- incorrect
 third parameter length, max 4."; log; fi
 w1=$(echo $1 | tr [a-z] [A-Z])
 w2=$(echo $2 | tr [a-z] [A-Z])
 w3=$(echo $3 | tr [a-z] [A-Z])
 figlet -f lean $w1 $w2 $w3 | sed -r 's/^[ ]*(_)/\1/'
```

```
Problemas shell_3
# Tenemos un laboratorio de PCs donde cada ordenador tiene un fichero
/etc/hosts que indica los nombres y direcciones IP de las demás
máquinas.
# Vamos a cambiar de dirección las máquinas gammaNN (donde NN es el
número del ordenador). La nueva dirección de cada máquina será
# 192.168.0.YY donde YY= NN+40
# Haz un script de shell que muestre por salida estándar el nuevo
fragmento de /etc/hosts.
grep 'gamma[0-9][0-9]' $1 | awk '
    {
        i=40 + substr($3,length($3)-2,length($3)-1)
        printf "192.168.0.%d %s %s\n",i,$2,$3
    }
    '
# Reducir tamaño de foto mediante:
# convert -geometry 800x600 origen destino
# Para publicar en el web, basta copiar al directorio public_html del
home del usuario. Suponemos este directorio existente y con
# los permisos adecuados. Proceden de un sistema contaminado por un
virus, así que hay ficheros que a pesar de su extensión, no son
# imágenes jpeg sino ejecutables. Si son verdaderas imágenes el comando
"file" mostrará un mensaje similar a este:
# imagen01.jpg: JPEG image data, EXIF standard 0.73, 10752 x 2048
# Haz un script de shell bash que reciba como primer argumento el
directorio, que compruebe cada fichero, que lo reduzca y publique
# si está bien y que lo borre si está contaminado, mostrando un mensaje
parecido a este:
#
# imagen01.jpg CONTAMINADO. Se borra el fichero
# imagen02.jpg ok. Reducida y publicada
# imagen03.jpg ok. Reducida y publicada
# 1 ficheros contaminados y borrados
# 2 ficheros reducidos y publicados
c=0; t=0
for i in $(ls $1)
do
    type=$(file $i | awk -F '[:,]' '{print $2}' | sed 's/^[ ]*//')
    if [ "$type" != "JPEG image data" ] && [ "$type" != "PNG image
data" ]
    then
        echo "$i CONTAMINADO. Se borra el fichero"
        c=$(($c + 1))
        #rm $i
    else
        echo "$i ok. Reducida y publicada"
        t=$(($t + 1))
        #convert -geometry 800x600 $i /$HOME/public_html
    fi
done
echo
echo "$c ficheros contaminados y borrados"
echo "$t ficheros reducidos y publicados"
```

```
 # Tenemos un directorio que contiene, entre otras cosas, scripts de
 shell. Se desea modificarlos, insertando entre su primera línea
 # y segunda el copyright del autor y la fecha.
 # Escribir un script de shell bash que haga esta tarea. Se usará por
 ejemplo de la siguiente forma:
 # pon_licencia $HOME/fuentes $HOME/licencia.txt
 if [ $# -ne 2 ]; then echo -e "\nusage: ./shell_tools_2.1.sh
 <files_directory> <licence_file> \n" >&2 ; exit 1; fi
 if [ ! -d $1 ]; then echo "$1 no es un directorio." >&2 ; exit 1; fi
 if [ ! -f $2 ]; then echo "$2 no es un fichero leible." >&2 ; exit 1;
 fi
 for i in $(ls $1)
 do
     if [ -f $i ]; then sed "1a# $(<$2)\n# $(date)\n" $i; fi
 done
```

```
# 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.
prefix=$1
d=$(date +%d%m%y)
shift
for dir in "$@"
do
    tar -cvzf ${prefix}_${d}.tar.gz $(find $dir -mtime +30 2>/dev/null)
done
```

```
comando < fichero    Toma la entrada de fichero
comando > fichero    Envía la salida de comando a
                     fichero; sobrescribe cualquier
                     cosa de fichero
comando 2> fichero   Envía la salida de error de
comando a fichero
comando << etiqueta  Añade la salida de comando al
final del fichero
comando 2>&1         Toma la entrada para comando
                     hasta que encuentra etiqueta en
                     alguna de las líneas
comando &> fichero   Envía la salida estándar y de
                     error a fichero; equivale a
                     comando > fichero 2>&1
comando1 | comando2 pasa la salida de comando1 a la
entrada de comando2

$0 el nombre del script
$1 a $9 parámetros del 1 al 9
${10}, ${11}, … parámetros a partir del 10
$# número de parámetros
$* todos los parámetros (como una secuencia)
$@ todos los parámetros (explícitamente separados)

$? Contiene el estado de salida (ejecución
correcta=0, o errónea != 0) del último comando (o
proceso) ejecutado
$$ Contiene el id del proceso en curso
$!Contiene el id del último proceso enviado como
tarea de fondo

# Formato de sustitución sed:
sed [opciones] s/REGEXP/reemplazo/flag [fichero]
# Flags:
   -  g aplica cambios globalmente
   -  p imprime las lineas afectadas
# Formato awk:
awk [[‘patron’] [{acción…}]…] ficheros…
awk -f fichero_comandos ficheros…

# useradd añade un nuevo usuario al sistema
-m crea directorio home
-g grupo al que pertenece
-s shell a usar
-e fecha de expiración de cuenta

# password permite cambiar o fijar contraseña
-e, --expire
-d, --delete
-l/-u, --lock/--unlock
-n, --mindays
-x, --maxdays
-w, --warndays
-i, --inactive
-S, --status

# comandos ssh:
ssh-keygen -t rsa/ed25519
ssh-copy-id [-i [identify_file]] [user@]machine
# Copy the file "foobar.txt" from a remote host
to the local host
$ scp your_username@remotehost.edu:foobar.txt
/some/local/directory


# iptables:
iptables -F chain-name
iptables -P chain-name target
iptables -A chain-name -i interface -j target
-p, --protocol
-s, --source
-d, --destination
-i, --interface
-o, --out-interface
-f, --fragment

# modprobe:
-r, --remove (descargar módulo)-C, --config (usa fichero
configfile)
-c, --showconfig (muestra ficheros config)
-n, --dry-run (no carga módulos)
-v, --verbose (imprime mensajes del programa)
```

# tipos de ficheros:			
regular	-	cp, cat…	rm
directorio	d	mkdir	rmdir
disp. char	c	mknod	rm
disp. blck	b	mknod	rm
socket	s	socket(2)	rm
pipe	p	mknod/mkfifo	rm
symb link	l	ln -s	rm

Paul Huszak
759934

```
# Para verificar que los usuarios de un sistema emplean contraseñas seguras,
realiza un script que busque en un nodo dado si los usuarios cuyo UID sea
mayor o igual a 1000 emplean contraseñas inseguras. El script recibirá como
argumentos una dirección IP y un fichero con las contraseñas inseguras a
probar, una por línea. Al terminar la búsqueda, el script enviara un correo
al usuario root del nodo donde se esté probando el script con a lista de
usuarios cuya contraseña haya sido encontrada. El asunto del correo será
“usuarios con mala contraseña”. Cuando los argumentos sean inválidos, el
script devolverá 1 y terminara escribiendo: “sintaxis invalida” por pantalla
# Notas:
#     • Se puede suponer que el usuario as puede acceder a cualquier nodo
mediante clave pública sin password.
#     • El comando sshpass permite escribir el password de ssh de manera no
interactiva. Una posible manera de utilización sería:
#          $ SSHPASS=<contraseña> sshpass -e ssh <usuario>@<nodo> <comando>
- El formato del fichero /etc/password es: Usuario:password
encriptado:UID.GUD
#
if [ $# -ne 2 ]; then echo "sintaxis invalida"; exit 1; fi
if [ ! -e $2 ]; then exit 1; fi
ssh as@$1 'logout'
if [ $? -ne 0 ]; then exit 1; fi
while read -r user
do
    while read -r passwd
    do
        SSHPASS=$passwd sshpass -e ssh ${user}@${$1} 'logout' &> dev/null
        if [ $? -eq 0 ]; then insecure_users+=($user); fi
    done < $2
done < <(ssh as@$1 "cat /etc/passwd | awk -F':' '{if(\$3 >= 1000) print
\$1}'")
printf "Node %s:\n %s\n" $1 ${insecure_users[@]} | mail -s "INSECURE USERS
IN NODE" root
```

```
# El primer conjunto de reglas inicializa la tabla
filter:
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
MUY IMPORTANTE: POR DEFECTO, SE RECHAZA TODO INPUT y FORWARD.
SOLO SE PERMITE LO QUE SE ESPECIFIQUE EXPLICITAMENTE.


# Permitir conexiones a los servicios de red (SSH,HTTP, HTTPS)
en 10.1.1.2
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 22 -j
ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 80 -j
ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 443 -j
ACCEPT
```

```
# NAT (Network Address Translation)
Mecanismo para remapear direcciones, útil para que varios nodos
dentro de una red privada puedan compartir una IP pública. NAT
oculta la organización de la red interna y previene el accedo a
los nodos desde el exterior.
DNAT (Destination NAT): cambia dirección destino del paquete
antes del routing.
SNAT (Source NAT): cambia dirección origen del paquete después
de routing.
MASQUERADE: tipo de SNAT usado cuando la asignación de IP es
dinámica.
```

```
# Comandos mount:
-Montaje: sudo mount -t ext3 /dev/sda4 /home
-Desmontaje/liberación: sudo umount /home
-Buscar Referencias: fuser -c /home
-Listar montajes: mount o cat /etc/mtab
-Listar dispositivos de bloque: lsblk

# procesos, comando ps:
-e o ax: visualizar todos los procesos
-u o U o --user <user>: visualizar procesos del usuario
-U <usuario>
-f o --forest: agrupa salida de forma jerárquica
O, --o <format>: especifica formato de salida
O <order> o --sort [k]<spec>: ordena la salida

# crontab:
minuto,hora,día,mes,día_semana,comando
crontab [-u usuario] fichero|{-l|-e|-r}
```

```
# 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.
#
lsmod | sed '1d' | awk '{if($3==0) print $1}' | while read -r module; do modprobe -r $module; done
```

```
# Configuración DHCP
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet dhcp
      hostname turza
# Configuración Estática
auto eth0
iface eth0 inet static
      address 192.168.0.3
      netmask 255.255.255.0
      broadcast 192.168.0.255
      network 192.168.0.0
      gateway 192.168.0.1
```

```
# 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.
ffld=0
sfld=1
while [ "$ip" != "155.155.255.254" ]; do
    ip=155.155.${ffld}.${sfld}
    ping $ip -i 0.1 -c 1 &> /dev/null
    if [ $? -eq 0 ]; then echo $ip;
    else
        if [ $sfld -eq 255 ]; then ffld=$(($ffld + 1)); sfld=0; fi
        sfld=$(($sfld + 1));
    fi
done | while read -r node; do nodes=$(($nodes + 1)); ssh -nf user@$node 'uname -r'; done | \
sort | uniq -c | awk -v "t=$nodes" 'BEGIN {printf "%d nodos linux".} 'printf "%s %d%",$2,$1/t'
```