Administrar cuentas de usuario y grupo y sus archivos relacionados

Peso	5
Tópico Cubierto	107.1 Administrar cuentas de usuarios y grupos y archivos relacionados
Descripción	Los alumnos deberán ser capaces de agregar, borrar y modificar usuarios y grupos.
Temas	Administrar información de la bases de datos y usuarios y grupos.
Ejemplos	* /etc/passwd
	* /etc/shadow
	* /etc/group
	* /etc/skel
	* usermod
	* useradd
	* userdel
	* groupadd
	* groupdel
	* groupmod
	* chage
	* passwd

Peso: Indica el valor de importancia que tiene este tópico en la certificación.

Tópico Cubierto: Indica, según el programa de certificación LPI, qué tópico le corresponde a este tema.

Descripción: Un resumen de lo que se verá.

Temas: Un resumen de los conceptos primordiales que están cubiertos.

Ejemplos: Palabras claves que se deben tener en cuenta.

Introducción

En este tópico se introducirán temas acerca de la administración de usuarios, se explicarán los comandos pertinentes y buenas prácticas para poder tener nuestro sistema perfectamente configurado de acuerdo a nuestras necesidades.

Tareas Administrativas

Administrando cuentas de usuario, grupos y ficheros del sistema

En este capítulo abordaremos temas como la creación, eliminación, suspensión y cambio en las cuentas de usuario del sistema. También aprenderemos a crear y eliminar grupos. Por último, estudiaremos la manera de cambiar el grupo al cual pertenece un usuario y anexarlo a otro. Comenzaremos hablando de los ficheros directamente relacionados con las cuentas de los usuarios y la función que estos desempeñan.

Archivo /etc/passwd

Este fichero almacena datos sobre las cuentas de usuario del sistema. Dichos datos se encuentran organizados línea a línea, de las cuales cada una corresponde a un usuario. También podemos encontrar algunos servicios del sistema como el servidor web o el de correo. Cada una de estas líneas se encuentra formada por 7 campos, los cuales se encuentran separados por el signo ":"

Los campos de los que hablamos están conformados por los siguientes parámetros:

1	UserName
2	X
3	UserID
4	GroupID
5	UserInfo
6	HomeDir
7	Shell

Descripción de los campos

1. Nombre de usuario
2. Es comúnmente conocido como el campo de "password" y se encuentra directamente ligado a la contraseña del usuario. Este campo puede adoptar 3 parámetros posibles:
    a) La letra '**x**' indica que el password del usuario se encuentra cifrado y ligado directamente al archivo **/etc/shadow**
    b) Si se encontrase este campo vacío, significa que el usuario no tiene contraseña.
3. Identificador de Usuario "UserID"
4. Identificador que indica a cuál grupo pertenece el usuario "GroupID"
5. Información adicional
6. Home del Usuario, directorio de trabajo del usuario
7. Intérprete de comandos asignado al usuario

Ejemplo

 # cat /etc/passwd

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

sync:x:4:65534:sync:/bin:/bin/sync

(...salida cortada...)

Archivo /etc/shadow

Este fichero almacena las contraseñas cifradas de cada usuario del sistema. Estos datos se encuentran organizados línea a línea, de las

cuales cada una corresponde a un usuario. Cada una de estas líneas se encuentra formada por nueve campos,que se encuentran separados por el signo "**:**"

Estos nueve campos se encuentran conformados por los siguientes parámetros

Campo	Descripción
1	Nombre de usuario
2	Contraseña cifrada Vacío=Sin password. El asterisco "*" o el signo de admiración "**!**" indican que el usuario no podrá loguearse en el sistema, esta opción es comúnmente asignada a usuarios que hacen uso de servicios como FTP. Algunas distribuciones como Fedora usan el doble signo de admiración "**!!**" cuando se crea un usuario y aun no se le ha asignado contraseña.
3	Días transcurridos desde el 1/ene/1970 hasta la fecha en que la contraseña fue cambiada por última vez. Nunca vacío.
4	Número de días que deben transcurrir hasta que la contraseña se pueda volver a cambiar. (0=Siempre permite cambio)
5	Número de días tras los cuales hay que cambiar la contraseña. (-1 significa nunca). A partir de este dato, se obtiene la fecha de expiración de la contraseña. Normalmente son 10000 días
6	Número de días antes de la expiración de la contraseña en que se le avisará al usuario al inicio de la sesión (Vacío=Sin advertencia)
7	Días después de la expiración en que la contraseña se inhabilitará, si es que no se cambio. (Vacío=nunca inactivo)
8	Fecha de caducidad de la cuenta. Se expresa en días transcurridos desde el 1/Enero/1970 vacío=Nunca será deshabilitado.
9	Reservado para uso futuro

Administración de cuentas de usuario

Las cuentas de usuario están localizadas en el fichero **/etc/passwd** y las contraseñas cifradas de los usuarios son asignadas al archivo **/etc/shadow**. Cuando una nueva cuenta de usuario es creada (usando el comando **useradd**), de manera predeterminada toma la plantilla (opción **-m**) **/etc/skel** para generar el entorno de trabajo del usuario (/home/nombredeusuario).

Comando useradd

Para generar una cuenta de usuario haremos uso del comando useradd, siguiendo esta sintaxis:

 # useradd [opciones] nombreDelUsuario

Opciones	Descripción
-b	Define la base para el home del usuario
-d	Define el home del usuario
-e	Se usa para especificar la fecha en la que expira la cuenta. Debe especificarse en el siguiente formato Año-Mes-Dia.
Ejemplos	-e 20100506
Ejemplos	-e 20081224
Ejemplos	-e 20090214
-f	Número de días antes de que la contraseña expire, o de que la cuenta sea deshabilitada
-g	El nombre del grupo o gid asignado a un nuevo usuario
-G	Grupo secundario al cual puede ser asignado un usuario
Ejemplos	-G desarrolloJava
Ejemplos	-G ventasMedicas
Ejemplos	-G soportePHP
-u	Identificador o uid que será asignado al usuario, por defecto Linux asignará UID's a partir del numero 500
-m	Crea el home del usuario
-s	Interprete de comandos SHELL que será asignado al usuario. Ej: /bin/bash

Ejemplo

 # useradd -g sistemas usuario1

Comando usermod

El comando **usermod** modifica los parámetros de acceso asignados a una cuenta existente del sistema.

Sintaxis:

 # usermod [-c comment] [-d home_dir [-m]] nombreDelUsuario

Opciones	**Descripción**
-c	Añade o modifica el comentario, campo 5 de /etc/passwd
-d	Modifica el directorio de trabajo o home del usuario, campo 6 de /etc/passwd
-e	Cambia o establece la fecha de expiración de la cuenta, formato AAAA-MM-DD, campo 8 de /etc/shadow
-g	Cambia el número de grupo principal del usuario (GID), campo 4 de /etc/passwd
-G	Establece otros grupos a los que puede pertenecer el usuario, separados por comas.
-l	Cambia el login o nombre del usuario, campo 1 de /etc/passwd y de /etc/shadow
-L	Bloquea la cuenta del usuario, no permitiéndole que ingrese al sistema. No borra ni cambia nada del usuario, solo lo deshabilita.
-s	Cambia el shell por defecto del usuario cuando ingrese al sistema.
-u	Cambia el UID del usuario.
-U	Desbloquea una cuenta previamente bloqueada con la opción -L.

Si quisiéramos cambiar el nombre de usuario de **'carita'** a **'carlita'**:

 # usermod -l carita carlita

Casi seguro también cambiará el nombre del directorio de inicio o HOME en /home, pero si no fuera así, hacemos lo siguiente:

 # usermod -d /home/carlita carlita

Otros cambios o modificaciones en la misma cuenta:

 # usermod -c “supervisor de área” -s /bin/ksh -g 505 carlita

Lo anterior modifica el comentario de la cuenta, su shell por defecto, que ahora será Korn shell, y su grupo principal de usuario que quedó establecido al GID 505. Todo esto se aplicó al usuario que, como se observa, debe ser el último argumento del comand

Comando userdel

El comando userdel remueve un usuario del sistema. Sintaxis:

 # userdel [opción] nombreDelUsuario

Opciones	**Descripción**
-r	Este parámetro indica que se elimina la cuenta y la carpeta de trabajo del usuario con **todos** sus datos. Si usáramos el comando userdel sin el parámetro **-r** ,solo eliminará al usuario del sistema
-f	Elimina todos los del usuario, cuenta, directorios y archivos, pero además lo hace sin importar si el usuario está actualmente en el sistema trabajando.

Comando passwd

El comando **passwd** se utiliza para cambiar contraseñas. Cuando se emplea el comando **passwd** sin opciones, se cambia la contraseña del usuario que lo invocó. Primero nos exigirá la contraseña vigente y luego pedirá dos veces la nueva para prevenir cualquier error. La utilización del comando **passwd** con los parámetros usuario y contraseña sólo es posible para **root**. Si se utiliza sólo el parámetro usuario cuando se hace uso de este comando, entonces **root** puede cambiar la contraseña para ese usuario.

Los caracteres admitidos para las contraseñas son los siguientes:

 # * , . ; : _ - + ! $ % & / | ? { [( )] }

Sintaxis:

 # passwd [Opciones] nombreDelUsuario

Opciones	**Descripción**
-e	Esta opción forzará al usuario a cambiar su contraseña en su siguiente login al sistema
-l	Con esta opción, el administrador del sistema puede inhabilitar la cuenta de algún usuario específico
-u	Con esta opción, el administrador del sistema puede deshabilitar la cuenta de algún usuario específico
-n	Mínimo número de días antes de poder cambiar
-x	Máximo número de días de validez; luego pide cambiar
-f	Cambia el nombre completo del usuario
-s	Cambia el SHELL del usuario

Archivo /etc/login.defs

En el archivo de configuración **/etc/login.defs** están definidas las variables que controlan los aspectos de la creación de usuarios y de los campos de shadow usadas por defecto. Algunos de los aspectos que controlan estas variables son:

```
   Número máximo de días que una contraseña es válida PASS_MAX_DAYS
   El número mínimo de caracteres en la contraseña PASS_MIN_LEN
   Valor mínimo para usuarios normales cuando se usa useradd UID_MIN
   El valor umask por defecto UMASK
   Si el comando useradd debe crear el directorio home por defecto
CREATE_HOME
```

Basta con leer este archivo para conocer el resto de las variables que son autodescriptivas y ajustarlas al gusto. Recuerden que se usarán, principalmente, al momento de crear o modificar usuarios con los comandos useradd y usermod.

Otros comandos de inicio de sesión del usuario

Opciones	Descripción
id -ng [usuario]	Muestra el grupo actual al cual pertenece el usuario.
groups [usuario]	Muestra todos los grupos al cual pertenece el usuario
id -nu [usuario]	Muestra el nombre del usuario conectado
echo $USER	Muestra el usuario actual
id -u	Muestra el ID del usuario actual
users	Muestra los usuarios logeados localmente en el sistema
who	Muestra los usuarios logeados localmente en el sistema
w	Muestra los usuarios logeados localmente en el sistema
finger -l usuario	Muestra los usuarios logeados local o remotamente en el sistema
lastlog	Muestra los últimos accesos al sistema. La lista incluye inicios, apagados y accesos

Administración de grupos

Comando groupadd

Para dar de alta grupos de trabajo en el sistema usaremos el comando **groupadd** el cual deberá ser aplicado según la siguiente sintaxis:

 # groupadd [opciones] nombreDelGrupo

Opciones	Descripción
-g	Define mediante un valor numérico el ID del grupo, este número no puede ser uno negativo.
-r	Define un grupo del sistema. Un grupo del sistema es aquel que tiene un número de identidad (GID) de grupo por debajo del número 500. Este particular GID es utilizado por los servicios del sistema como un servidor web o de correo.
-f	Forza al sistema a crear el grupo aunque éste ya exista.
-o	Asigna un ID existente a un grupo.

Comando groupmod

El comando **groupmod** permite modificar el nombre o GID de un grupo. Sintaxis:

 # groupmod [-g nuevo-gid] [-n nuevoNombre] nombreDelGrupo

Opciones	Descripción
-g	Esta opción cambia el GID de un grupo existente en el sistema
-n	Esta opción sirve para cambiar el nombre de un grupo existente por otro

Comando groupdel

El comando **groupdel** elimina un grupo del sistema, su sintaxis es la siguiente:

 # groupdel nombreDelGrupo

Comando gpasswd

Permite administrar los grupos. Se puede utilizar para añadir y eliminar usuarios, señalar un administrador e indicar una contraseña para el grupo. NOTA: Las contraseñas de grupo sólo son necesarias si un usuario que no es miembro del mismo quisiera anexarse al grupo y convertirlo en uno de sus grupos efectivos, para ello deberá proporcionar la contraseña del grupo.

Sintaxis:

 # gpasswd [opciones] nombreDelGrupo

Opciones	Descripción
-R	Hace que el grupo sea reservado para miembros
-A usuario,, grupo	Señala como administrador de un grupo particular a un usuario del grupo
-M usuario,, grupo	Añade miembros a un grupo
-r grupo	Elimina la contraseña del grupo

Opciones para el administrador del grupo

Opciones	Descripción
-a usuario,, grupo	Se añade permanentemente un usuario a un grupo
-d usuario,, grupo	Se borra permanentemente a un usuario del grupo
-r grupo	Elimina la contraseña del grupo

Comando grpck

El comando **grpck** revisa un grupo de sistema Sintaxis:

 # grpck nombreDelGrupo

Archivo /etc/group

Este archivo guarda la relación de los grupos a los que pertenecen los usuarios del sistema y contiene una línea para cada usuario con tres o cuatro campos por usuario:

Ejemplo:

```
   El campo 1 indica el usuario.
   El campo 2 'x' indica la contraseña del grupo, que no existe, si hubiera
se mostraría un 'hash' encriptado.
   El campo 3 es el Group ID (GID) o identificación del grupo.
   El campo 4 es opcional e indica la lista de grupos a los que pertenece
el usuario
```

Actualmente, al crear al usuario con **useradd**, se crea también automáticamente su grupo principal de trabajo GID, con el mismo nombre del usuario. Es decir, si se añade el usuario 'paola', también se crea el **/etc/group** el grupo 'paola'.

Archivo /etc/gshadow

Este fichero almacena las contraseñas cifradas de los grupos, los administradores de cada grupo y los usuarios que pertenecen a cada grupo y contiene una línea para cada grupo con cuatros campos por grupo:

Ejemplo:

NombreDelGrupo:(Contraseña o !):listaDeAdministradores:listaDeMiembros

Cada uno de estos campos separados por el identificador "**:**" La lista de administradores y miembros de un grupo deben estar separados mediante el signo "**:**"

Bibliografía

Libros:

LPI Linux Certification in a Nutshell, Third Edition, June 2010 —> Capitulo 7

LPIC-1: Linux Professional Institute Certification Study Guide: (Exams 101 and 102), 2nd Edition, February

Páginas:
[1] Usuarios

Realizar tareas de administración de seguridad

Peso	3
Tópico Cubierto	110.1 Realizar tareas de administración de seguridad
Descripción	Los candidatos deberán saber como revisar la configuración del sistema para asegurar la seguridad del mismo de acuerdo a las políticas locales.
Temas	* Auditar el sistema para encontrar archivos con el bit suid/sgid establecidos * Establecer o cambiar la contraseña de usuarios y la información de expiración de las mismas. * Establecer límites en el login de usuario, procesos y uso de memoria. * Uso y configuración básica de sudo
Ejemplos	* find * chage * /etc/sudoers * visudo * su * ulimit * fuser * who, w, last

Peso: Indica el valor de importancia que tiene este tópico en la certificación.

Tópico Cubierto: Indica, según el programa de certificación LPI, qué tópico le corresponde a este tema.

Descripción: Un resumen de lo que se verá.

Temas: Un resumen de los conceptos primordiales que están cubiertos.

Ejemplos: Palabras claves que se deben tener en cuenta.

Introducción

En este tópico se realizarán tareas básicas para poder aplicar una seguridad básica y mínima al control de los usuarios, viendo temas como seguridad en claves, aplicación y creación reglas para ejecución de comandos administrativos,control de los recursos del sistema y auditar permisos especiales.

Seguridad en el equipo

La seguridad en el equipo es muy importante, no importa lo pequeña que parezca la tarea a securizar, lo importante es saber los métodos y aplicarlos según corresponda.

Comando chage

Se usa para listar o cambiar el tiempo en el que expira una contraseña de usuario

Sintaxis:

 # chage [opciones] nombreDelUsuario

Opciones	**Descripción**
-d días	Cuenta el número de días (desde 01-01-1970) transcurridos desde que cambió la contraseña por última vez. Se puede usar /MM/DD/YY

-E fecha Modifica la fecha en que la cuenta del usuario expirará y será bloqueada. Se puede usar /MM/DD/YY

-I días Modifica cuantos días puede permanecer una cuenta con una contraseña expirada antes de ser bloqueada

-M días Modifica el número máximo de días durante los que es válida la contraseña de usuario. Pasados los días, el usuario deberá de modificarla

-m días Modifica el número mínimo de días entre cambio de contraseña. Evita que el usuario cambie la clave reiteradas veces en el dia.

-W días Modifica el número de días que se avisará al usuario antes de cambiar la contraseña

-l usuario Muestra información del usuario especificado

Ejemplo

```
[root@oc6127656113 ~]# chage -l matias
Last password change                                    : Nov 26, 2011
Password expires                                        : Feb 24, 2012
Password inactive                                       : Mar 02, 2012
Account expires                                         : never
Minimum number of days between password change          : 5
Maximum number of days between password change          : 90
Number of days of warning before password expires       : 7
[root@oc6127656113 ~]#
```

Ahora cambiaremos cada uno de los ítems descriptos para ver cómo quedarían:

```
[root@oc6127656113 ~]# chage -d 10 -E 01/22/2012 -I 9 -M 5 -m 2 -W 2
matias
```

Teniendo en cuenta lo explicado arriba, cambiamos cada uno de sus parámetros:

```
[root@oc6127656113 ~]# chage -l matias
Last password change                                    : Jan 11, 1970
Password expires                                        : Jan 16, 1970
Password inactive                                       : Jan 25, 1970
Account expires                                         : Jan 22, 2012
Minimum number of days between password change          : 2
Maximum number of days between password change          : 5
Number of days of warning before password expires       : 2
[root@oc6127656113 ~]#
```

Es importante establecer estos puntos, dado que nos servirán para poder controlar bien el comportamiento de nuestras cuentas.

Si quisiéramos omitir todo tipo de seguridad:

```
[root@oc6127656113 ~]# chage -d -1 -E -1 -I -1 -M -1 -m -1 -W -11 matias
```

Quedaría así:

```
[root@oc6127656113 ~]# chage -l matias
Last password change                                    : never
Password expires                                : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : -1
Maximum number of days between password change          : -1
Number of days of warning before password expires       : -1
[root@oc6127656113 ~]#
```

O también así:

```
[root@oc6127656113 ~]# chage -d 999999 -E 999999 -I 999999 -M 999999 -m
999999 -W 999999 matias
```

Como verán, los tiempos que marcan son imposibles:

```
[root@oc6127656113 ~]# chage -l matias
Last password change                                    : Nov 28, 4707
Password expires                                : never
Password inactive                                       : never
Account expires                                         : Nov 28, 4707
Minimum number of days between password change          : 999999
Maximum number of days between password change          : 999999
Number of days of warning before password expires       : 999999
[root@oc6127656113 ~]#
```

Auditando Archivos

Otra tarea importante es la de chequear aquellos archivos que contengan permisos especiales SUID, SGID y sticky bit.

Estos tipos de permisos pueden llevar a que se ejecuten programas que no deberían tener accesos algunos usuario o grupos.

Buscar archivos con SUID activo

 # find / -type f -perm +4000 2>/dev/null

/usr/sbin/pppd

/usr/bin/X

/usr/bin/gpasswd

/usr/bin/chfn

/usr/bin/lppasswd

/usr/bin/passwd

/usr/bin/pkexec

/usr/bin/newgrp

/usr/bin/fping

/usr/bin/chsh

/usr/bin/sudo

(...salida cortada...)

Es lógico que algunos comandos figuren con este bit activado, ya que facilitan la administración de ciertas tareas, como las del comando **passwd**, que permite que cada usuario cambie su password.

Si quisiéramos buscarlo de otra forma:

 # find / -type f -perm -u=s -ls 2>/dev/null

```
7012450  992 -rwsr-xr-x   1 root     root       1011444 Aug 16  2013 /usr/sbin/vmware-authd
7012372  300 -rwsr-xr--   1 root     dip         302176 Jun 22  2012 /usr/sbin/pppd
9179027   12 -rwsr-sr-x   1 root     root          9508 May 11  2013 /usr/bin/X
9175219   72 -rwsr-xr-x   1 root     root         66196 May 25  2012 /usr/bin/gpasswd
9175223   44 -rwsr-xr-x   1 root     root         44564 May 25  2012 /usr/bin/chfn
9180127   16 -rwsr-xr-x   1 root     lpadmin      13712 Sep 29  2013 /usr/bin/lppasswd
9175220   48 -rwsr-xr-x   1 root     root         45396 May 25  2012 /usr/bin/passwd
```

Sgid

Ahora vamos corroborar aquellos archivos que estén afectados por el SGID.

 # find / -type f -perm -g=s -ls 2>/dev/null

```
6922941  132 -rwxr-sr-x   1 root     ssh         128396 Apr  2  2014 /usr/bin/ssh-agent
9175960   20 -rwxr-sr-x   1 root     tty          18020 Dec  9  2012 /usr/bin/wall
9177095   36 -rwxr-sr-x   1 root     crontab      34760 Jul  3  2012 /usr/bin/crontab
6923211   48 -rwsr-sr-x   1 daemon   daemon           46556 Jun  9  2012 /usr/bin/at
6922446  408 -rwxr-sr-x   1 root     utmp        410688 Sep 16  2012 /usr/bin/screen
9175222   56 -rwxr-sr-x   1 root     shadow           49364 May 25  2012 /usr/bin/chage
```

(...salida cortada...)

Otro ejemplo:

 # find / -type f -perm +2000 2>/dev/null

```
/usr/bin/ssh-agent
/usr/bin/wall
/usr/bin/crontab
/usr/bin/at
/usr/bin/screen
/usr/bin/chage
```

(...salida cortada...)

Ambos

Si quisiéramos buscar por ambos:

 # find / -type f -perm +6000 2>/dev/null

```
/usr/sbin/pppd
/usr/bin/X
/usr/bin/gpasswd
/usr/bin/ssh-agent
/usr/bin/chfn
/usr/bin/wall
/usr/bin/passwd
```

(...salida cortada...)

Comando fuser

El comando fuser se utiliza para identificar procesos utilizando archivos o sockets.

Sintaxis

 # fuser [opciones] archivo

Opciones

-k Envía SIGKILL a al proceso que está accediendo el archivo definido.

-u Muestra el nombre de usuario asociado al proceso.

-v Modo verboso

Ejemplo

Mostrar procesos y usuario asociados al bash

 # fuser /bin/bash

/usr/bin/bash: 1187e(root) 2275e(root)

La letra que se ve a continuación del PID es el tipo de acceso que se tiene al archivo.

Las letras más comunes son:

e El ejecutable está corriendo

F El archivo está abierto en modo escritura (solo en modo verboso)

f El archivo está abierto.

Comando w

El comando **w** muestra la cantidad de usuarios conectados, el tipo que lleva iniciado el sistema y la carga promedio.

 # w

23:17:08 up 6:33, 3 users, load average: 0,00, 0,02, 0,05

USER	TTY	LOGIN@	IDLE	JCPU	PCPU	WHAT
root	tty2	16:48	4.00s	8.69s	0.09s	w
root	tty3	18:43	4:33m	0.41s	0.41s	-bash

Opción	**Descripción**
LOGIN	Hora de conexión
IDLE	Tiempo ocioso
JCPU	Tiempo de uso que llevan todos los procesos en la tty definida.
PCPU	Tiempo usado por el proceso actual mostrado en la columna WHAT

Comando who

El comando **who** muestra quien se encuentra conectado en el sistema

 # who

root	tty2	2014-12-11 16:44
root	tty3	2014-12-11 12:14

Utilizando Ulimit

Para poder hacer un uso correcto de los recursos de nuestro equipo, a veces tenemos que asignar políticas un poco más restrictivas, en el uso de los recursos.

1. ¿Qué sucedería si tuviéramos que limitar el acceso ?

2. ¿Qué sucedería si necesitamos mostrar la cantidad de procesos y tamaño de archivo que un usuario pudiese usar?
3. ¿Qué sucedería si quisiéramos limitar por tiempo el uso del procesador?
4. ¿Qué sucedería si quisiéramos limitar la cantidad de archivos a abrir ?
5. ¿Qué sucedería si quisiéramos limitar el tamaño de un archivo?
6. ¿Qué sucedería si quisiéramos limitar el tamaño de memoria Ram que ocupa un programa?
7. ¿Qué sucedería si quisiéramos limitar la prioridad de un proceso?
8. Estas serían algunas preguntas y, a continuación, veremos sus respuestas.

Con ulimit podremos limitar esto, entre otras cosas

1. Para implementar esto lo hacemos directamente en el archivo **/etc/profile** para que cada vez que un usuario acceda, tenga su perfil general con sus restricciones.
2. Otra manera de realizarlo es mediante los módulos de PAM (pam_limits), editando el archivo **/etc/security/limits.conf** y teniendo en cuenta que el módulo debe estar activado correctamente.

Funcionamiento de Ulimit

Veamos los parámetros fundamentales de ulimit para luego configurarlo con alguno de los métodos anteriormente descriptos.
Es importante saber que los límites que estableceremos solo aplican a la terminal donde lo ejecutamos o desde donde se ejecutan si usan pam_limits o el /etc/profile.
Los recursos que podremos limitar se dividen en las siguientes categorías:

Recurso	Descripción	Opción
core	Restringe el tamaño de los core file	-c (blocks)
data	Restringe el tamaño que ocupan los programas en el area RAM (con cero no hace core dumps)	-d (KB)
fsize	Restringe el tamaño de los archivos creados por el usario	-f (KB)
nofile	Restringe la cantidad de archivos que un usuario puede abrir	-n
stack	Restrige el tamaño del stack	-s (KB)
cpu	Restringe el uso del procesador para un proceso	-t (minuto)
nproc	Restringe la cantidad de procesos que un usuario puede tener	-u
maxlogins	Setea la cantidad de veces que un usuario puede estar logueado	
priority	Setea la prioridad que va utilizar un proceso de un usuario	
locks	Setea la cantidad máxima de files locks	
nice	Setea la prioridad máxima permitida para modificar un proceso	
Hard	Setea/Muestra los límites Hard	-H
Soft	Setea/Muestra los límites Soft	-S
Listar	Lista todos los estados	-a

Estos límites tienen un límite **Soft** y un límite **Hard** que permite que se sobrepase el soft, siempre y cuando no supere el hard si es que algún proceso o recursos requieren más que el límite Soft.
Para ver los límites Soft podemos hacer: (puede hacerlo un usuario sin privilegios)

$ ulimit -a -S

core file size (blocks, -c) 0

```
data seg size       (kbytes, -d)        unlimited
scheduling priority             (-e)    0
file size           (blocks, -f)        unlimited
pending signals                 (-i)    31457
max locked memory   (kbytes, -l)    64
max memory size     (kbytes, -m)    unlimited
open files                      (-n)    1024
pipe size           (512 bytes, -p)     8
POSIX message queues (bytes, -q)            819200
real-time priority              (-r)    0
stack size          (kbytes, -s)        8192
cpu time            (seconds, -t)       unlimited
max user processes              (-u)    31457
virtual memory      (kbytes, -v)        unlimited
file locks                      (-x)    unlimited
```

Muestra limites Hard:

$ ulimit -a -H

```
core file size      (blocks, -c)        unlimited
data seg size       (kbytes, -d)        unlimited
scheduling priority         (-e)                0
file size           (blocks, -f)        unlimited
pending signals                 (-i)    31457
max locked memory   (kbytes, -l)    64
max memory size     (kbytes, -m)    unlimited
open files                      (-n)    4096
pipe size           (512 bytes, -p)     8
POSIX message queues (bytes, -q)            819200
real-time priority              (-r)    0
stack size          (kbytes, -s)        16384
cpu time            (seconds, -t)       unlimited
max user processes              (-u)    31457
virtual memory                  (kbytes, -v)        unlimited
file locks                      (-x)    unlimited
```

Como podemos ver, entre paréntesis nos indica con qué opción podemos configurar cada recurso.

Ejemplos

Si quisiéramos modificar alguno de los límites tan solo bastaría mirar la tabla y poner uno nuevo, siempre y cuando este adentro de los rangos permitidos. Esto depende de pam_limits y la configuración de /etc/profile , sino podríamos setear límites ilimitados.

Vamos a limitar la cantidad máxima de procesos del usuario en su límite SOFT:

$ ulimit -Su 234

```
[ariel@restaurador ~]$ ulimit -a |grep "max user"
max user processes              (-u) 234
[ariel@restaurador ~]$
[ariel@restaurador ~]$ ulimit -Ha |grep "max user"
max user processes              (-u) 912
```

El límite se redujo en 234 y el límite HARD quedo como estaba.

Al intentar modificar el hard:

```
[ariel@restaurador ~]$ ulimit -Hu 1024
-bash: ulimit: max user processes: cannot modify limit: Operation not
permitted
[ariel@restaurador ~]$
```

No permite modificar el límite HARD debido a que superamos el límite que teníamos asignado.

Definir el tamaño máximo por archivo

```
[ariel@restaurador ~]$ ulimit -aH |grep "file size"| grep -v core
file size               (blocks, -f) unlimited
[ariel@restaurador ~]$
```

Definir un tamaño de soft de un bloque y hard de 4 bloques

```
Agregamos estas líneas en el /etc/profile
ulimit -S -f 1
ulimit -H -f 4
Veamos ahora:
[ariel@restaurador ~]$ ulimit -aH |grep "file size"| grep -v core
file size               (blocks, -f) 4
[ariel@restaurador ~]$ ulimit -aS |grep "file size"| grep -v core
file size               (blocks, -f) 1
[ariel@restaurador ~]$
[ariel@restaurador ~]$ dd if=/dev/zero of=edudacionit.avi bs=1024 count=6
File size limit exceeded
[ariel@restaurador ~]$
[ariel@restaurador ~]$ dd if=/dev/zero of=edudacionit.avi bs=1024 count=1
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied, 7.2736e-05 s, 14.1 MB/s
[ariel@restaurador ~]$
```

Al intentar crear un archivo de 6kb muestra error, ya que supera el tamaño definido en el límite HARD. En cambio al crear un archivo de 1kb, no hay problemas.

Vamos a limitar la cantidad de procesos que pueda usar:

```
[ariel@restaurador ~]$ ulimit -Su 3
[ariel@restaurador ~]$ ulimit -Hu 6
[ariel@restaurador ~]$ ulimit -Ha |grep -i processes
max user processes              (-u) 6
[ariel@restaurador ~]$ ulimit -Sa |grep -i processes
max user processes              (-u) 3
[ariel@restaurador ~]$
```

Generemos un pequeño script, el cual creará montones de procesos.

```
[ariel@restaurador ~]$ vi test
[ariel@restaurador ~]$ chmod u+x test
[ariel@restaurador ~]$ cat test
```

```
#!/bin/bash
while true; do echo "hola" >> /tmp/hola; sleep 3; done
[ariel@restaurador ~]$
```

```
[ariel@restaurador ~]$ ./test &
[1] 13419
[ariel@restaurador ~]$ ./test &
-bash: fork: retry: Resource temporarily unavailable
..
..
```

Configurando con /etc/profile

Para aplicar **ulimit**, si es que no utilizamos PAM, podemos agregar en **/etc/profile** los comandos según los usuarios y lo que necesitemos.

En la sección que más les guste o armándose un script más prolijo, pueden agregar todas las sintaxis correspondientes.

```
ulimit -S -u 512
ulimit -H -u 912
ulimit -S -f 1
ulimit -H -f 4
```

Configurando limits.conf

Para poder usar el **/etc/security/limits.conf**, que va de la mano del módulo de PAM pam_limits, deberíamos tener los siguientes archivos.

```
[root@restaurador ~]# grep -i limits /etc/pam.d/*
/etc/pam.d/fingerprint-auth:session     required      pam_limits.so
/etc/pam.d/password-auth:session     required      pam_limits.so
/etc/pam.d/runuser:session              requiredpam_limits.so
/etc/pam.d/smartcard-auth:session     required      pam_limits.so
/etc/pam.d/sudo:session    required     pam_limits.so
/etc/pam.d/sudo-i:session    required     pam_limits.so
/etc/pam.d/system-auth:session     required      pam_limits.so
[root@restaurador ~]#
```

```
[root@restaurador ~]# ls -l /etc/security/limits.conf
-rw-r--r-- 1 root root 1825 dic  7 22:20 /etc/security/limits.conf
```

También podemos definir otros límites de forma más modular.

```
[root@restaurador ~]# ls -l /etc/security/limits.d/
total 4
-rw-r--r-- 1 root root 152 dic  7 22:20 90-nproc.conf
[root@restaurador ~]#
```

Ejemplo:

Defino para el usuario1 un tamaño máximo de archivo de 400kb

/etc/security/limits.d/prueba

usuario1 hard fsize 400

El corazón de la configuración está dentro del siguiente archivo:

 # cat /etc/security/limits.conf

```
#<domain> can be:
#        - an user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax, for maxlogin limit
#        - NOTE: group and wildcard limits are not applied to root.
#        To apply a limit to the root user, <domain> must be the literal username root.
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit (KB)
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues (bytes)
#        - nice - max nice priority allowed to raise to values: [-20, 19]
#        - rtprio - max realtime priority
#        - chroot - change root to directory (Debian-specific)
#*               soft    core            0
#root            hard    core            100000
#*               hard    rss     10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#@student        -       maxlogins       4
```

Para entender cómo se usa este archivo hay que explicar cada categoría:

#<domain> <type> <item> <value>

Domain: Puede tomar los siguientes valores

- nombre de usuario
- nombre de grupo, con @grupo
- comodín *, para valor por defecto
- comodín %, se puede usar también para los grupos, %grupo, también para indicar máximo login.

Type: Puede tener dos valores

- soft para este tipo de límite
- hard para este tipo de límite

Ítem: Se especifica qué tipo de acción corresponde al item

- core
- data
- fsize
- memlock
- nofile
- rss
- stack
- cpu
- nproc
- as
- maxlogins
- maxsyslogins
- priority
- locks
- sigpending
- msgqueue
- nice
- rtprio

Value: El valor que tomará cada ítem

Un ejemplo para tomar puede ser el que utilizamos arriba, que varía según lo que se necesite en cada ocasión. Lo ideal es saber que está ahí y usarlo cuando sea necesario.

Un ejemplo para ver lo que las aplicaciones están consumiendo(virtual size vs rss):?

Ordenado de forma ascendente por virtual size, podría pasar que el virtual size es mayor que el residen size y ahí hay que corregirlo.

```
while read rss vsz command; do rss="$(bc <<< "scale=2; ${rss}/1024")";
vsz="$(bc <<< "scale=2; ${vsz}/1024")"; echo "${rss} ${vsz} $command";
done < <(ps --no-headers -A -o rss,vsz,comm --sort vsz)
```

Anexo: Fork Bomb

Este ejemplo limita la cantidad máxima de procesos concurrentes para que se evite la famosa fork bomb que nos colapsa el sistema con un usuario sin privilegios.

```
[root@restaurador ~]# cat /etc/security/limits.d/90-nproc.conf
# Default limit for number of user's processes to prevent
# accidental fork bombs.
# See rhbz #432903 for reasoning.

*          soft    nproc     1024
[root@restaurador ~]#
```

Fork Bomb:

```
:(){ :|:& };:
```

Explicación:

```
/- Define the function ':' without any parameters '()' as follows:
 | /- Beginning of function-block.
 | | /- Load a copy of the function ':' into memory ...
 | | |/- ... and pipe its output to ...
 | | ||/- ... another copy of the ':'-function, which has to be loaded
into memory.
 | | |||   (In other words, ':|:' loads two more copies of ':', thus
causing a chain-reaction)
 | | |||/- Disown the functions (make them a background process), so that
the children of a parent
 | | ||||  will not be killed when the parent gets auto-killed.
 | | |||| /- End of function-block.
 | | |||| |/- End of definition.
/-\| |||| ||/- Execute the function ':'.  The chain-reaction begins.
:(){ :|:& };:
```

Código:

```
forkbomb(){ forkbomb|forkbomb & } ; forkbomb
```

En POSIX o C:

```
#include <unistd.h>

int main()
{
  while(1)
    fork();
  return 0;
}
```

nologin

Para que nadie pueda acceder al equipo, salvo root, basta con crear el archivo **/etc/nologin**. Si quisiéramos habilitarlo, tendríamos que habilitar el módulo en PAM.

```
[root@restaurador etc]# grep -i nologin /etc/pam.d/*
/etc/pam.d/login:account    required     pam_nologin.so
/etc/pam.d/remote:account    required     pam_nologin.so
/etc/pam.d/samba:auth       required    pam_nologin.so
```

```
/etc/pam.d/sshd:account    required     pam_nologin.so
[root@restaurador etc]#
```

Utilizando sudo/su

Con estos dos tipos de comandos podremos ejecutar comandos sin necesidad,en algunos casos, de tipear la password de root. La idea es ejecutar esos comandos sin la necesidad de tener que loguearnos como root.

Usando su

Con el comando **su** podemos ejecutar comandos sencillos sin necesidad de acceder como root, pero ingresando su clave.
Como verán, en este ejemplo ejecutamos un comando que lista los socket abiertos, pero como usuario normal no veremos nada, asi que lo ejecutaremos con **su**.

Sintaxis

$ su [opciones] usuario

Opciones

-c comando	Ejecuta el comando previo el ingreso de la contraseña (si no se especifica usuario, utiliza root)
-l	Se identifica en la terminal (en vez de "**-l**" se puede poner solo "**-**")
-m	Preserva las variables de entorno (se puede utilizar "**-p**")

$ lsof -i

$ su -c 'lsof -i'

Password:

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
rsync	3888	root	4u	IPv4	9393	0t0	TCP	*:rsync (LISTEN)
rsync	3888	root	5u	IPv6	9394	0t0	TCP	*:rsync (LISTEN)
sshd	3989	root	3u	IPv4	8082	0t0	TCP	*:ssh (LISTEN)
sshd	3989	root	4u	IPv6	8084	0t0	TCP	*:ssh (LISTEN)
vsftpd	4013	root	3u	IPv4	9423	0t0	TCP	*:ftp (LISTEN)

(...salida cortada...)

Cambiar de usuario sin privilegios a **root**

$ su -

Password:

 # whoami

root

Usando sudoers

Aplicando políticas de **sudoers** podremos delegar la administración a determinados usuarios o grupos para evitar tener que darles el password de root. De esta manera, cada sector o persona aplica una serie de reglas que van a determinar qué es lo que pueden ejecutar delegándole "permisos de root".
Para eso, tendremos que utilizar la herramienta **visudo** que abrirá el archivo **/etc/sudoers** con el editor de texto que tengamos configurado; también podemos abrirlo con un editor predilecto, aunque es preferible utilizar visudo, porque chequea la sintaxis y evita que varios escriban al mismo tiempo.
En resumen con sudoers podemos aplicar plantillas para definir quién hace qué y hasta desde dónde puede hacerlo.

Sintaxis

Directiva	Descripción
User_Alias	Especifica una lista de usuarios que permitirá ejecutar una directiva
Cmnd_Alias	Especifica una lista de comandos que ciertos usarios pueden ejecutar
Host_Alias	Especifica una lista de equipos en donde los usuarios podrán ejecutar las directivas
Runas_Alias	Especifica comandos que pueden correr como determinado usuario

User_Alias

Agrupa usuarios en un “alias” identificador

```
##
# User alias specification
##
User_Alias      FULLTIMERS = millert, mikef, dowdy
User_Alias      PARTTIMERS = bostley, jwfox, crawl
User_Alias      WEBMASTERS = will, wendy, wim
```

Runas_Alias

Agrupa usuarios para que luego una o más aplicaciones puedan ser ejecutadas como alguno de los usuarios definidos

```
##
# Runas alias specification
##
Runas_Alias     OP = root, operator
Runas_Alias     DB = oracle, sybase
```

Host_Alias

Agrupa en un “alias” identificador un grupo de máquinas por ip o nombre. Esta directiva solo se utiliza en servidores centralizados por NIS o LDAP

```
##
# Host alias specification
##
Host_Alias      SPARC = bigtime, eclipse, moet, anchor:\
                SGI = grolsch, dandelion, black:\
                ALPHA = widget, thalamus, foobar:\
                HPPA = boa, nag, python
Host_Alias      CUNETS = 128.138.0.0/255.255.0.0
Host_Alias      CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
Host_Alias      SERVERS = master, mail, www, ns
Host_Alias      CDROM = orion, perseus, hercules
```

Cmnd_Alias

Agrupa en un “alias” identificatorio a un grupo de comandos

```
##
# Cmnd alias specification
##
Cmnd_Alias      DUMPS = /usr/sbin/dump, /usr/sbin/rdump,
/usr/sbin/restore, \
                        /usr/sbin/rrestore, /usr/bin/mt
Cmnd_Alias      KILL = /usr/bin/kill
Cmnd_Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias      SHUTDOWN = /usr/sbin/shutdown
Cmnd_Alias      HALT = /usr/sbin/halt, /usr/sbin/fasthalt
Cmnd_Alias      REBOOT = /usr/sbin/reboot, /usr/sbin/fastboot
Cmnd_Alias      SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                         /usr/local/bin/tcsh, /usr/bin/rsh, \
                         /usr/local/bin/zsh
Cmnd_Alias      SU = /usr/bin/su
Cmnd_Alias      VIPW = /usr/sbin/vipw, /usr/bin/passwd, /usr/bin/chsh, \
                       /usr/bin/chfn
```

Ejemplos

Habilitar a usuario1 y usuario2 a reiniciar la PC

 # visudo

User_Alias USUARIOS_REINICIO = usuario1, usuario2, @grupo

Cmnd_Alias REINICIAR = /sbin/init 6, /sbin/reboot, /sbin/shutdown -r

USUARIOS_REINICIO ALL=REINICIAR

usuario HOST=(como_quien) commando

Habilitar todos los comandos excepto uno, en este caso el **su**

 # visudo

Cmnd_Alias SU = /usr/bin/su

usuario1 ALL = ALL,!SU

Habilitar los usuarios que pertenecen al grupo **wheel** a ejecutar cualquier comando con **sudo** sin limitaciones.

 # visudo

%wheel ALL = (ALL) ALL

Habilitar un usuario para ejecutar todo, pero que no le pida contraseña

 # visudo

usuario1 ALL=(ALL) NOPASSWD:ALL

Listando Reglas

Listar las reglas activas por usuario

```
[restauracion@/home/crond1 $] sudo -l
Matching Defaults entries for crond1 on this host:
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR
LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS
LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
env_keep+="LC_TIME LC_ALL
    LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User crond1 may run the following commands on this host:
    (root) NOPASSWD: /usr/lib/jupiter/scripts/bluetooth, (root)
/usr/lib/jupiter/scripts/cpu-control, (root)
/usr/lib/jupiter/scripts/resolutions, (root)
    /usr/lib/jupiter/scripts/rotate, (root)
/usr/lib/jupiter/scripts/touchpad, (root) /usr/lib/jupiter/scripts/vga-
out, (root) /usr/lib/jupiter/scripts/wifi
    (ALL) ALL
[02:36:26]
[restauracion@/home/crond1 $]
```

Bibliografía

Libros:

LPI Linux Certification in a Nutshell, Third Edition, June 2010

LPIC-1: Linux Professional Institute Certification Study Guide: (Exams 101 and 102), 2nd Edition, February 2009

Anexo 1: Comando pwconv y pwunconv

El comportamiento por defecto de todas las distros modernas de GNU/Linux es activar la protección extendida del archivo **/etc/shadow**, que (se insiste) oculta efectivamente el 'hash' cifrado de la contraseña de **/etc/passwd**

Pero si por alguna situación extraña de compatibilidad se requiriese tener las contraseñas cifradas en el mismo archivo de **/etc/passwd** se usaría el comando **pwunconv**:

 # cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
hack:x:501:500:Hack Beastie:/home/hack:/bin/bash
(...salida cortada...)

La '**x**' en el campo 2 indica que se hace uso de **/etc/shadow** como mencionamos anteriormente.

 # cat /etc/shadow

root:ghy675gjuXCc12r5gt78uuu6R:10568:0:99999:7:7:-1::
hack:rfgf886DG778sDFFDRRu78asd:10568:0:-1:9:-1:-1::
(...salida cortada...)

 # pwunconv

 # cat /etc/passwd

root:ghy675gjuXCc12r5gt78uuu6R:0:0:root:/root:/bin/bash
hack:rfgf886DG778sDFFDRRu78asd:501:500:Hack Beastie:/home/hack:/bin/bash

En cualquier momento es posible reactivar la protección de shadow:

 # pwconv

 # ls -l /etc/passwd /etc/shadow

-rw-r–r– 1 root root 1106 2007-07-08 01:07 /etc/passwd
-r——— 1 root root 699 2009-07-08 01:07 /etc/shadow

Se vuelve a crear el archivo shadow, además nótese los permisos tan restrictivos (400) que tiene este archivo, haciendo sumamente difícil que cualquier usuario que no sea root lo lea.

Anexo 2: Comando grpconv y grpunconv

Estos comandos funcionan de la misma manera que **pwconv** y **pwunconv**, solo que aplicados a los ficheros

```
/etc/group
/etc/gshadow
```

Anexo 3: Comando newusers

La mayoría de las veces que creemos nuevos usuarios, utilizaremos el comando **useradd**, pero cuando tenemos una red de muchos usuarios o tenemos que crear o actualizar múltiples usuarios en un servidor se hace útil poderlo hacer desde una lista, para esto, podemos utilizar el comando **newusers**. El comando **newusers** toma un archivo de texto plano que deberá tener el mismo formato que el de el archivo **/etc/passwd** de nuestro sistema. Al ejecutar el comando y darle como parámetro la ruta al archivo de texto, creará los usuarios no existentes y actualizará los que ya existen; así mismo, si no existe la carpeta /home/nombreDelUsuario, la creará por nosotros. Para hacer uso de este comando haremos lo siguiente: Primero creamos el archivo de texto con los usuarios:

]# touch /root/batch-nuevos-usuarios.txt

 # chmod 0600 /root/batch-nuevos-usuarios.txt

Es importante que este archivo sólo pueda ser leído por root (por eso el chmod 0600) ya que las claves en este archivo estarán en texto plano.

Agregar la lista de usuarios, recuerden que debe tener el mismo formato que **/etc/passwd**:

usuario1:password:1001:513:Cuenta Mercadeo:/home/usuario1:/bin/bash
usuario2:password:1002:513:Usuario Ventas:/home/usuario2:/bin/bash
pepe:password:1110:501:Cuenta Invitado:/home/guest:/bin/menu

Agreguen cuantos usuarios quieran, igualmente podemos agregar los usuarios que quieran actualizar. (muy útil para hacer un reset de password masivo).

Por último, ejecutar el comando:

 # newusers /root/batch-nuevos-usuarios.txt