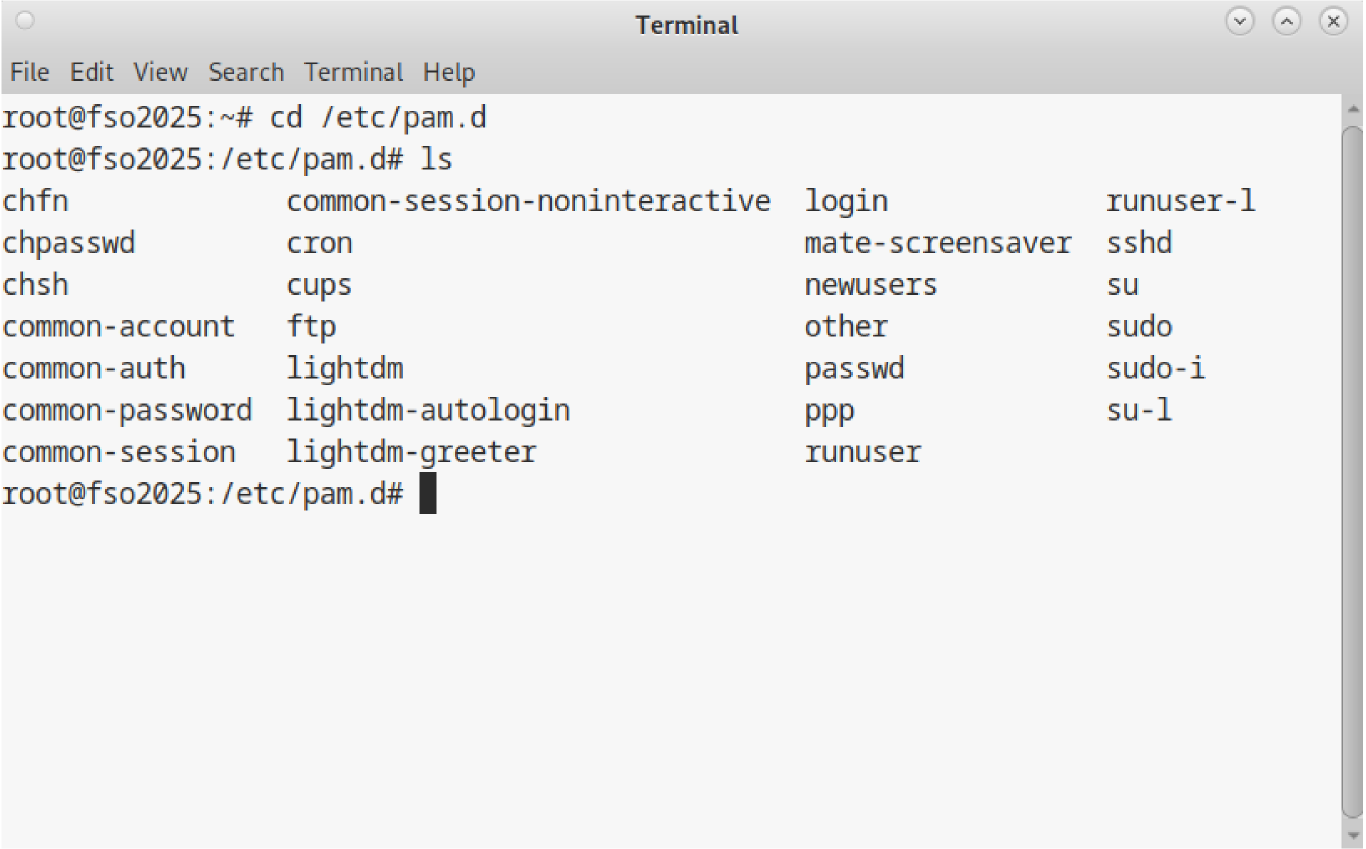
[FORT] Práctica 4: Securizando las cuentas de usuario

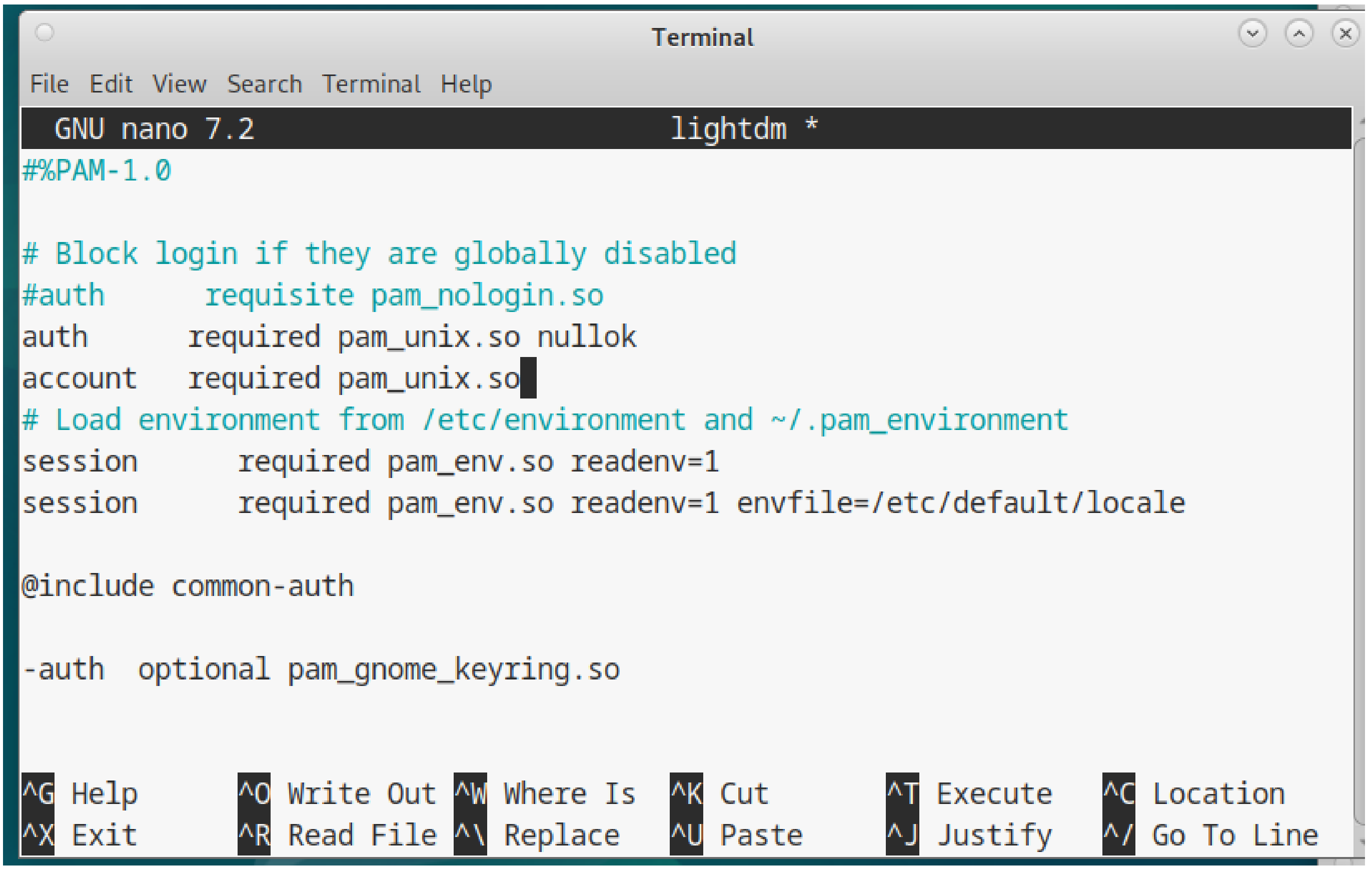
1. Deshabilita el login a root, tanto en el Display Manager como en las Terminales Virtuales excepto tty3



```
root@fso2025:~# cd /etc/pam.d
root@fso2025:/etc/pam.d# ls
chfn             common-session-noninteractive  login            runuser-l
chpasswd         cron                           mate-screensaver sshd
chsh             cups                           newusers         su
common-account   ftp                            other            sudo
common-auth      lightdm                        passwd           sudo-i
common-password  lightdm-autologin              ppp              su-l
common-session   lightdm-greeter                runuser
root@fso2025:/etc/pam.d# 
```

Para deshabilitar estos permisos vamos a modificar los archivos de PAM que se encuentran en /etc/pam.d. Para deshabilitar el acceso a root en la interfaz gráfica modificamos el archivo lightdm con las siguientes lineas:

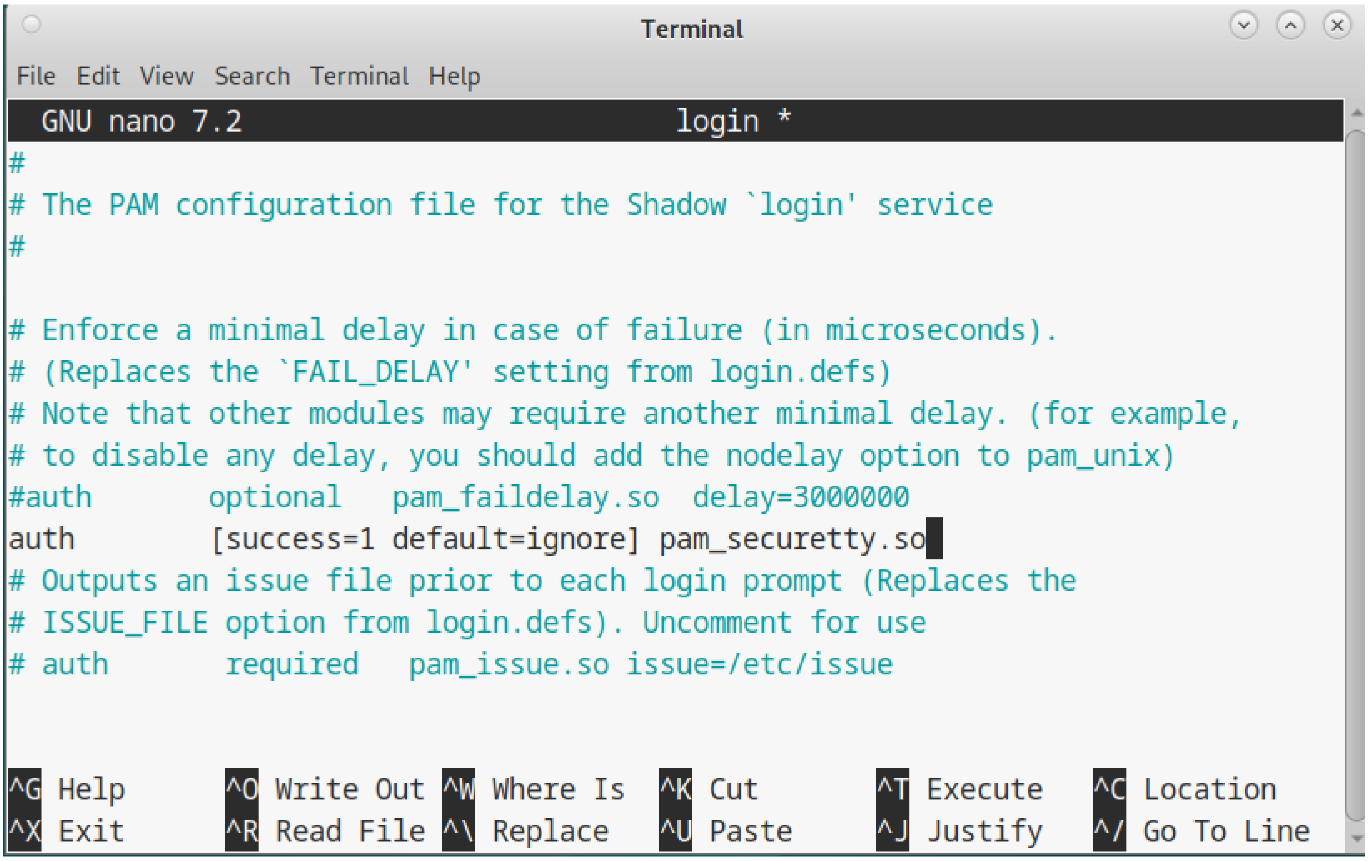
```
auth required pam_unix.so nullok
account required pam_unix.so
```



```
#%PAM-1.0

# Block login if they are globally disabled
#auth      requisite pam_nologin.so
auth      required pam_unix.so nullok
account   required pam_unix.so
# Load environment from /etc/environment and ~/.pam_environment
session      required pam_env.so readenv=1
session      required pam_env.so readenv=1 envfile=/etc/default/locale

@include common-auth

-auth  optional pam_gnome_keyring.so


^G Help       ^O Write Out ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File ^\ Replace   ^U Paste      ^J Justify    ^/ Go To Line
```

Para deshabilitar los permisos en las terminales virtuales modificamos el archivo login con lo siguiente:

```
auth [success=1 default=ignore] pam_securetty.so
```



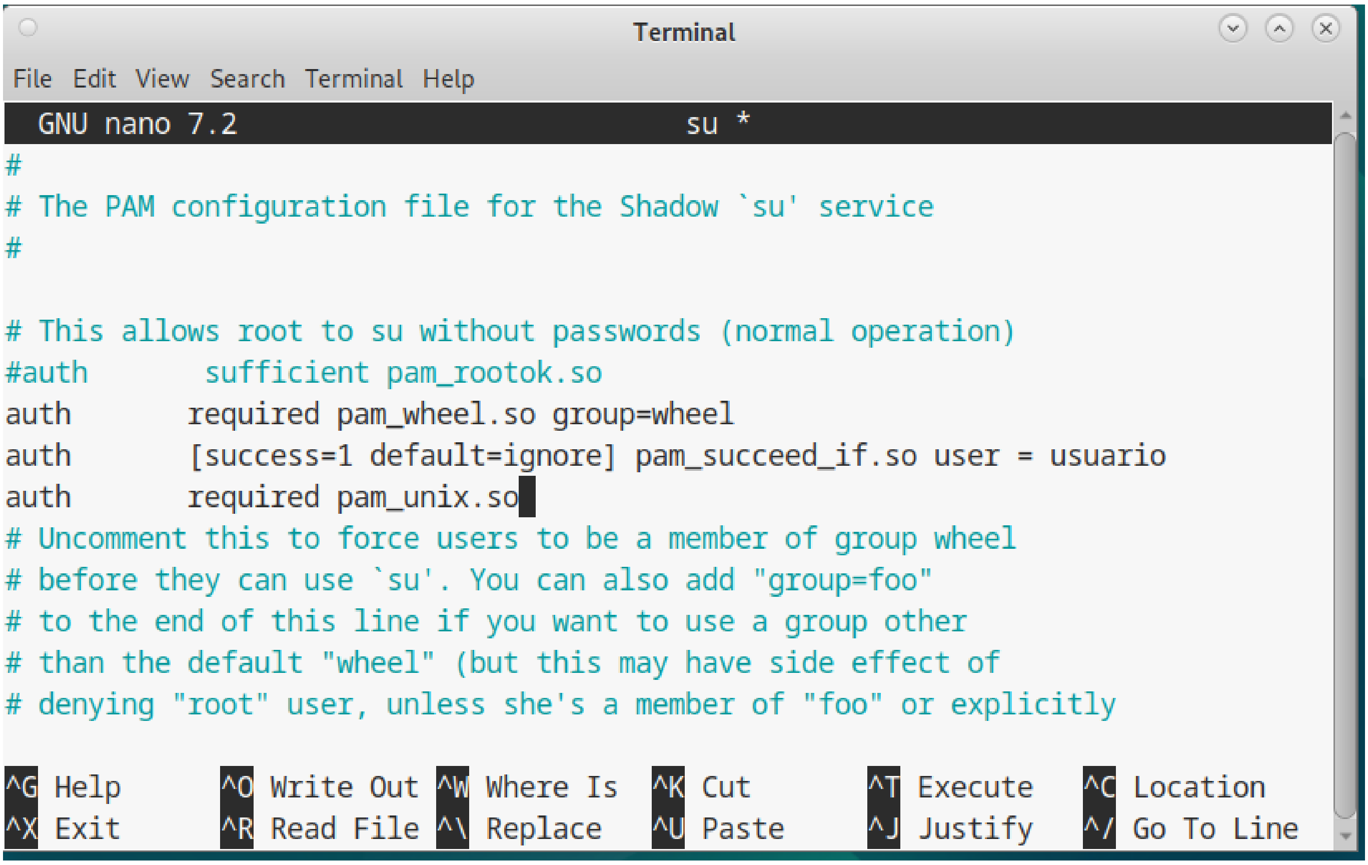
```
  GNU nano 7.2                          login *
#
# The PAM configuration file for the Shadow `login' service
#

# Enforce a minimal delay in case of failure (in microseconds).
# (Replaces the `FAIL_DELAY' setting from login.defs)
# Note that other modules may require another minimal delay. (for example,
# to disable any delay, you should add the nodelay option to pam_unix)
#auth       optional   pam_faildelay.so  delay=3000000
auth        [success=1 default=ignore] pam_securetty.so
# Outputs an issue file prior to each login prompt (Replaces the
# ISSUE_FILE option from login.defs). Uncomment for use
# auth       required   pam_issue.so issue=/etc/issue


^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

2. Usando el módulo pam_wheel.so haz que solo usuario, user001, user002, user003 y user004 puedan volverser root con SU.

Usuario no necesita saber la contraseña, mientras que los usuarios del 001 al 004 la necesitan para convertirse en root. Al resto de usuarios no se les preguntará por la contraseña.

Para aplicar los ajustes debemos modificar el archivo /etc/pam.d/su con las siguientes líneas:

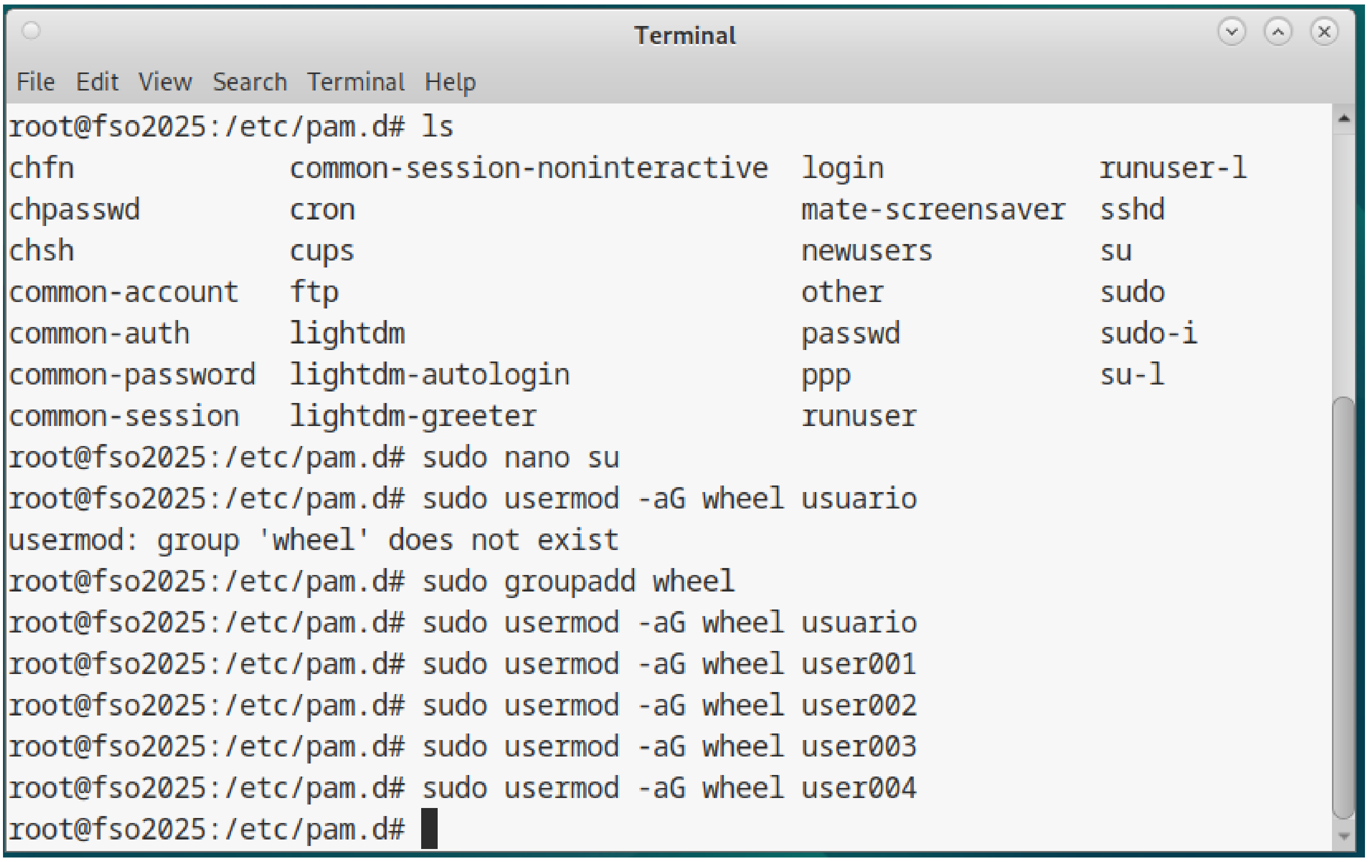
```
auth required pam_wheel.so group=wheel #Permite a los usuario del grupo Wheel a usar su sin contraseña
auth [success=1 default=ignore] pam_succeed_if.so user = usuario #Permite al usuario "usuario" usar su sin contraseña
auth required pam_unix.so #se configura para permitir que los usuarios indicados tengan que poner la contraseña
```



```
#
# The PAM configuration file for the Shadow `su' service
#

# This allows root to su without passwords (normal operation)
#auth       sufficient pam_rootok.so
auth       required pam_wheel.so group=wheel
auth       [success=1 default=ignore] pam_succeed_if.so user = usuario
auth       required pam_unix.so
# Uncomment this to force users to be a member of group wheel
# before they can use `su'. You can also add "group=foo"
# to the end of this line if you want to use a group other
# than the default "wheel" (but this may have side effect of
# denying "root" user, unless she's a member of "foo" or explicitly
```

También debemos añadir a los usuarios en cuestión al grupo wheel con el siguiente comando:

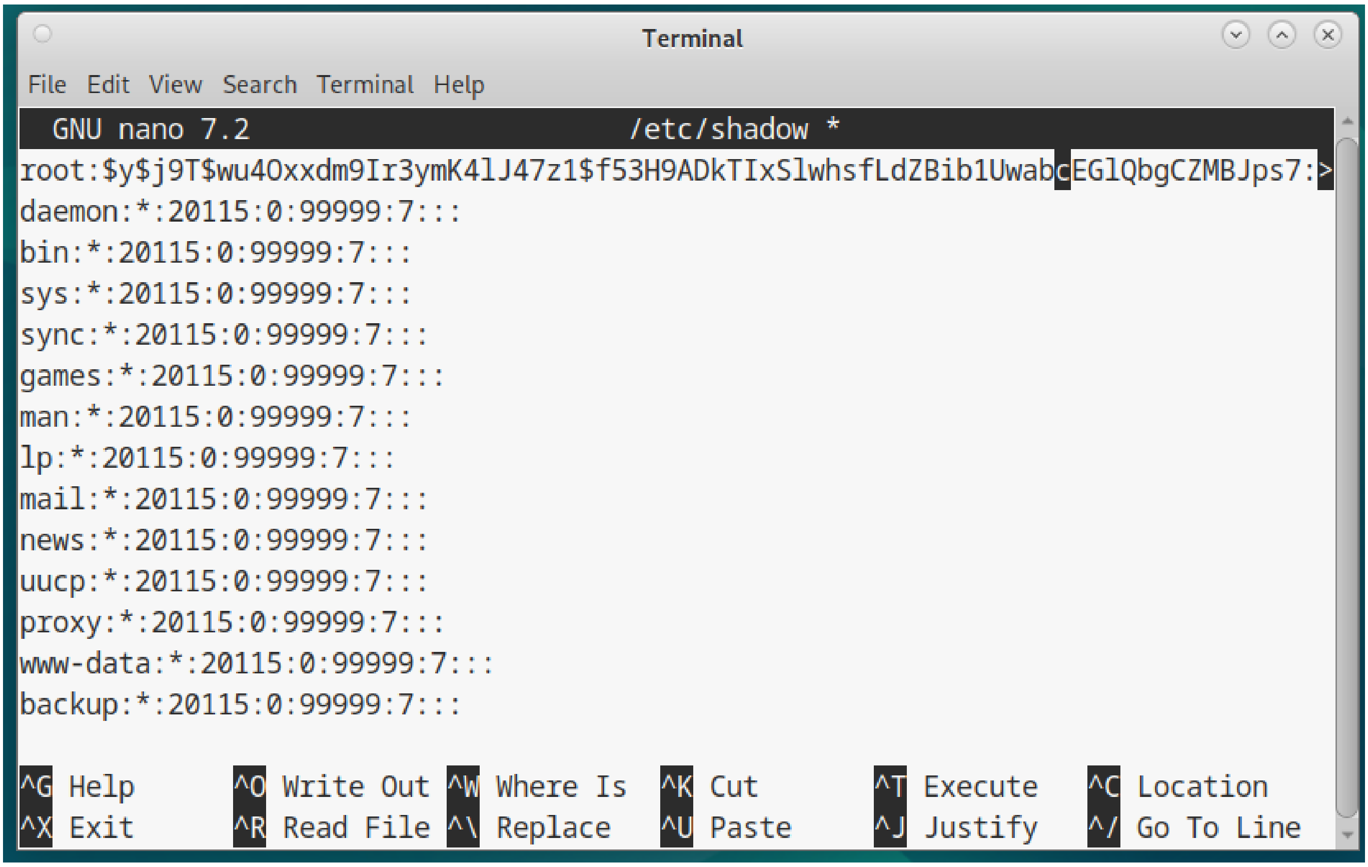
```
sudo usermod -aG wheel usuario
sudo usermod -aG wheel user001
sudo usermod -aG wheel user002
sudo usermod -aG wheel user003
sudo usermod -aG wheel user004
```



```
root@fso2025:/etc/pam.d# ls
chfn             common-session-noninteractive  login            runuser-l
chpasswd         cron                           mate-screensaver sshd
chsh             cups                           newusers         su
common-account   ftp                            other            sudo
common-auth      lightdm                        passwd           sudo-i
common-password  lightdm-autologin              ppp              su-l
common-session   lightdm-greeter                runuser
root@fso2025:/etc/pam.d# sudo nano su
root@fso2025:/etc/pam.d# sudo usermod -aG wheel usuario
usermod: group 'wheel' does not exist
root@fso2025:/etc/pam.d# sudo groupadd wheel
root@fso2025:/etc/pam.d# sudo usermod -aG wheel usuario
root@fso2025:/etc/pam.d# sudo usermod -aG wheel user001
root@fso2025:/etc/pam.d# sudo usermod -aG wheel user002
root@fso2025:/etc/pam.d# sudo usermod -aG wheel user003
root@fso2025:/etc/pam.d# sudo usermod -aG wheel user004
root@fso2025:/etc/pam.d# 
```

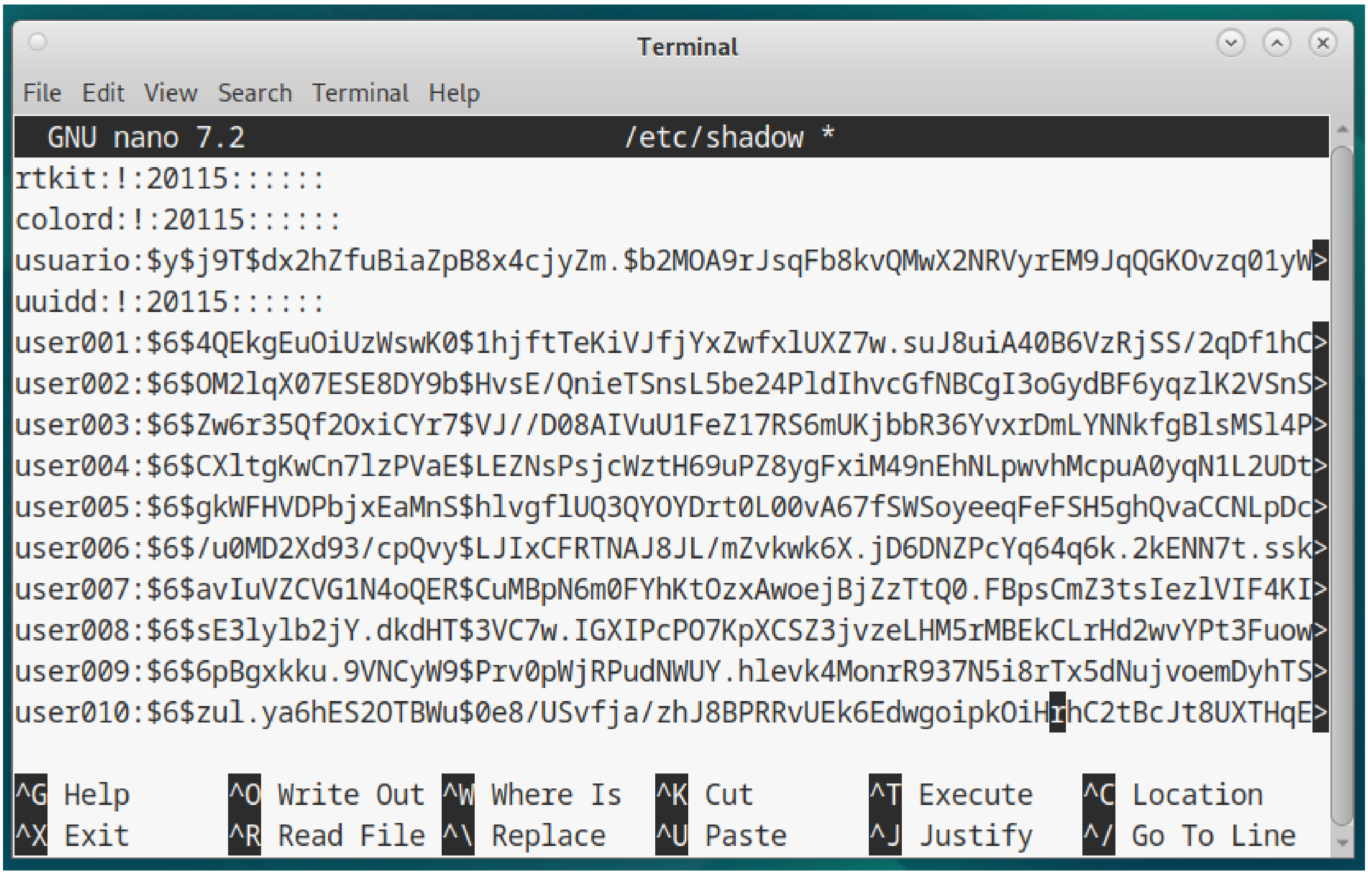
3. ¿Que método de cifrado se usa para las contraseñas? ¿Ha sido usado el mismo método para todas las contraseñas en el sistema?

Deberíamos usar SHA256 y cambiar las contraseñas de todos los usuarios a SHA256 **¿Como deberíamos hacer?** Echando un vistazo a /etc/shadow podemos ver las contraseñas cifradas por un lado para root:



```
  GNU nano 7.2                     /etc/shadow *
root:$y$j9T$wu4Oxxdm9Ir3ymK4lJ47z1$f53H9ADkTIxSlwhsfLdZBib1UwabcEGlQbgCZMBJps7:>
daemon:*:20115:0:99999:7:::
bin:*:20115:0:99999:7:::
sys:*:20115:0:99999:7:::
sync:*:20115:0:99999:7:::
games:*:20115:0:99999:7:::
man:*:20115:0:99999:7:::
lp:*:20115:0:99999:7:::
mail:*:20115:0:99999:7:::
news:*:20115:0:99999:7:::
uucp:*:20115:0:99999:7:::
proxy:*:20115:0:99999:7:::
www-data:*:20115:0:99999:7:::
backup:*:20115:0:99999:7:::

^G Help        ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit        ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

y por otro lado para los demás usuarios:

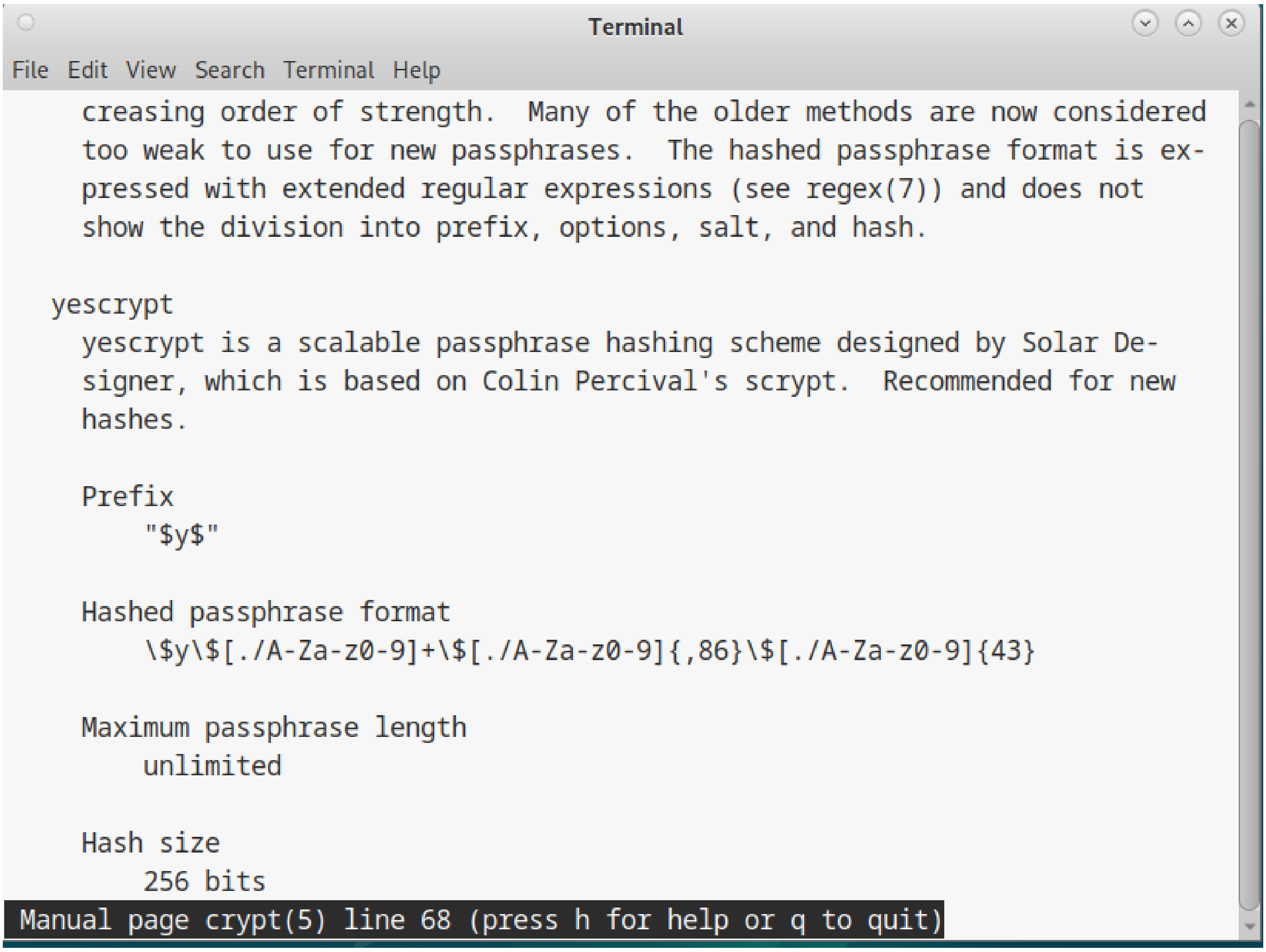


```
  GNU nano 7.2                     /etc/shadow *
rtkit:!:20115::::::
colord:!:20115::::::
usuario:$y$j9T$dx2hZfuBiaZpB8x4cjyZm.$b2MOA9rJsqFb8kvQMwX2NRVyrEM9JqQGKOvzq01yW>
uuidd:!:20115::::::
user001:$6$4QEkgEuOiUzWswK0$1hjftTeKiVJfjYxZwfxlUXZ7w.suJ8uiA40B6VzRjSS/2qDf1hC>
user002:$6$OM2lqX07ESE8DY9b$HvsE/QnieTSnsL5be24PldIhvcGfNBCgI3oGydBF6yqzlK2VSnS>
user003:$6$Zw6r35Qf2OxiCYr7$VJ//D08AIVuU1FeZ17RS6mUKjbbR36YvxrDmLYNNkfgBlsMSl4P>
user004:$6$CXltgKwCn7lzPVaE$LEZNsPsjcWztH69uPZ8ygFxiM49nEhNLpwvhMcpuA0yqN1L2UDt>
user005:$6$gkWFHVDPbjxEaMnS$hlvgflUQ3QYOYDrt0L00vA67fSWSoyeeqFeFSH5ghQvaCCNLpDc>
user006:$6$/u0MD2Xd93/cpQvy$LJIxCFRTNAJ8JL/mZvkwk6X.jD6DNZPcYq64q6k.2kENN7t.ssk>
user007:$6$avIuVZCVG1N4oQER$CuMBpN6m0FYhKtOzxAwoejBjZzTtQ0.FBpsCmZ3tsIezlVIF4KI>
user008:$6$sE3lylb2jY.dkdHT$3VC7w.IGXIPcPO7KpXCSZ3jvzeLHM5rMBEkCLrHd2wvYPt3Fuow>
user009:$6$6pBgxkku.9VNCyW9$Prv0pWjRPudNWUY.hlevk4MonrR937N5i8rTx5dNujvoemDyhTS>
user010:$6$zul.ya6hES2OTBWu$0e8/USvfja/zhJ8BPRRvUEk6EdwgoipkOiHrhC2tBcJt8UXTHqE>

^G Help        ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit        ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

Las contraseñas en general comienzan con una cadena \$<caracter>\$, estos primeros 3 caracteres señalan en que cifrado viene cada contraseña, por lo que sabemos que:

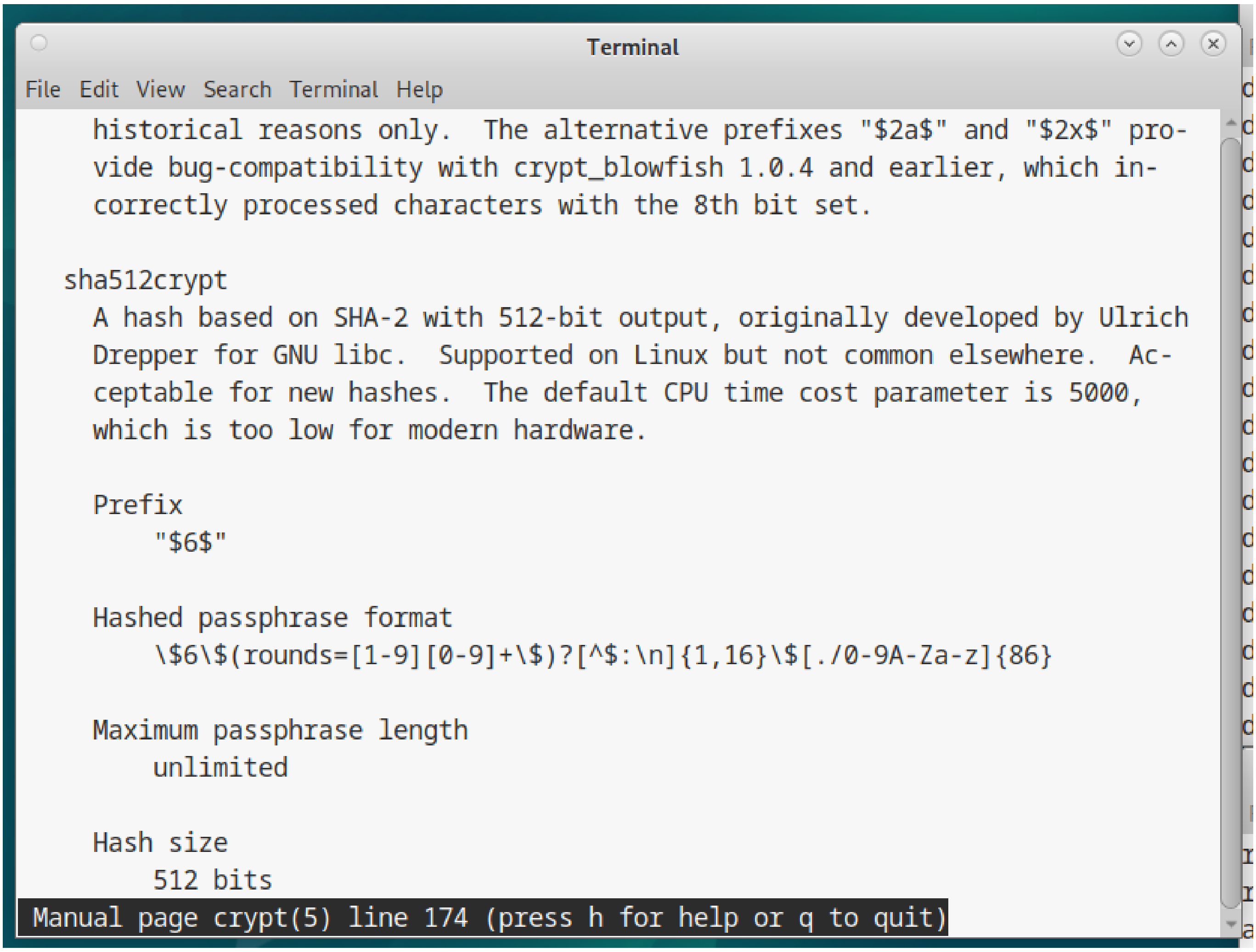
- Root y usuario comienzan con \$y\$, por lo que sabemos que usa yescrypt
- El resto de usuarios comienzan con \$6\$, por lo que sabemos que usan SHA-512

Esto lo sabemos gracias a las salidas del comando man 5 crypt, donde podemos ver que es cada cifrado, por ejemplo, el de root y usuario sería el siguiente:



```
     creasing order of strength.  Many of the older methods are now considered
     too weak to use for new passphrases.  The hashed passphrase format is ex-
     pressed with extended regular expressions (see regex(7)) and does not
     show the division into prefix, options, salt, and hash.

   yescrypt
     yescrypt is a scalable passphrase hashing scheme designed by Solar De-
     signer, which is based on Colin Percival's scrypt.  Recommended for new
     hashes.

     Prefix
         "$y$"

     Hashed passphrase format
         \$y\$[./A-Za-z0-9]+\$[./A-Za-z0-9]{,86}\$[./A-Za-z0-9]{43}

     Maximum passphrase length
         unlimited

     Hash size
         256 bits
 Manual page crypt(5) line 68 (press h for help or q to quit)
```

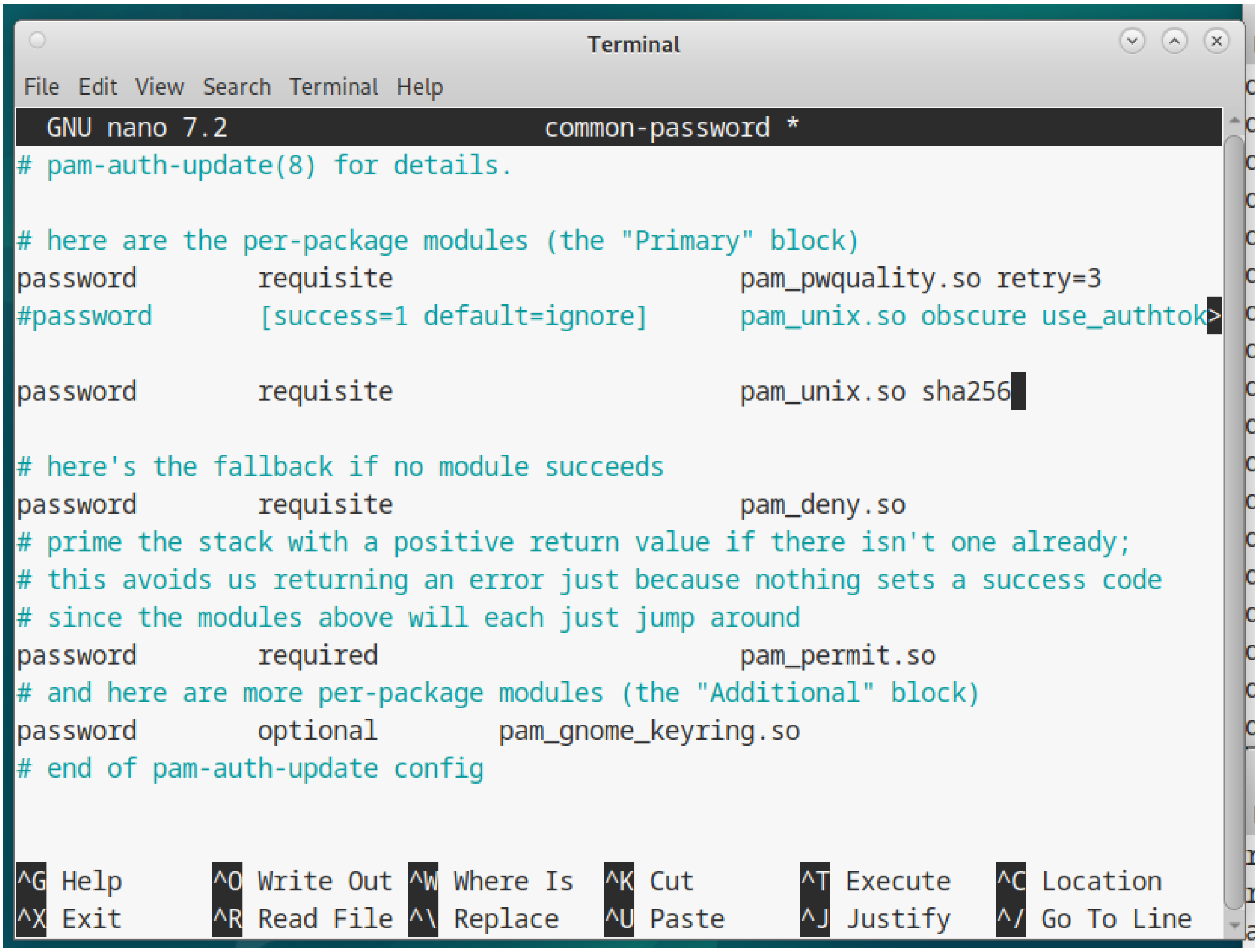
Y el del resto de usuarios sería el siguiente:



```
       historical reasons only.  The alternative prefixes "$2a$" and "$2x$" pro-
       vide bug-compatibility with crypt_blowfish 1.0.4 and earlier, which in-
       correctly processed characters with the 8th bit set.

   sha512crypt
       A hash based on SHA-2 with 512-bit output, originally developed by Ulrich
       Drepper for GNU libc.  Supported on Linux but not common elsewhere.  Ac-
       ceptable for new hashes.  The default CPU time cost parameter is 5000,
       which is too low for modern hardware.

       Prefix
           "$6$"

       Hashed passphrase format
           \$6\$(rounds=[1-9][0-9]+\$)?[^$:\n]{1,16}\$[./0-9A-Za-z]{86}

       Maximum passphrase length
           unlimited

       Hash size
           512 bits
 Manual page crypt(5) line 174 (press h for help or q to quit)
```

Para cmbiar el cifrado de todos los usuarios a SHA256 habría que primero cambiar cual es el cifrado predeterminado en /etc/pam.d/common-password con las siguientes líneas:

```
password requisite pam_unix.so sha256
```



```
  GNU nano 7.2                     common-password *
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
password        requisite                       pam_pwquality.so retry=3
#password       [success=1 default=ignore]      pam_unix.so obscure use_authtok>

password        requisite                       pam_unix.so sha256

# here's the fallback if no module succeeds
password        requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password        required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
password        optional        pam_gnome_keyring.so
# end of pam-auth-update config


^G Help        ^O Write Out  ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit        ^R Read File  ^\ Replace   ^U Paste      ^J Justify    ^/ Go To Line
```

Y tras eso modificar todas las contraseñas con un script como este:

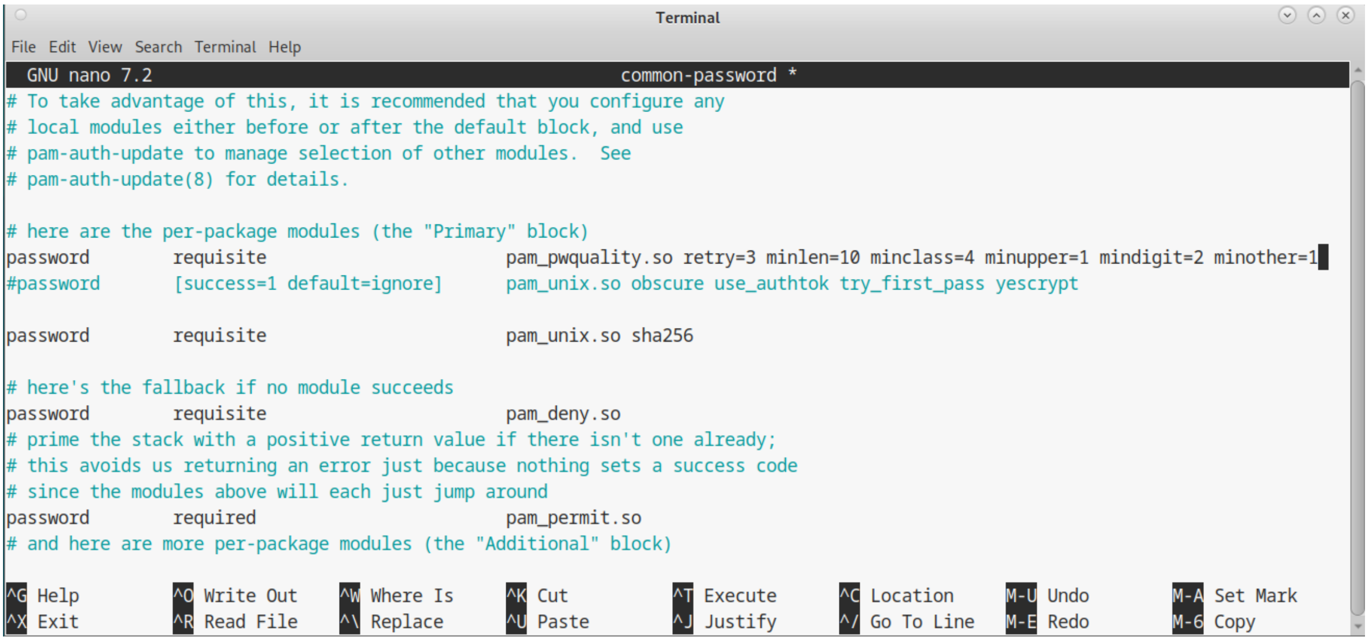
```
for user in $(cut -f1 -d: /etc/passwd); do
  sudo passwd --stdin $user
done
```

4. Fuerza los siguientes requisitos para los cambios de contraseña

- **Al menos 10 caracteres**
- **Debe contener mayúsculas y minúsculas**
- **Debe contener al menos 2 dígitos**
- **Debe contener al menos un caracter no alphanumérico**
- **No puede ser una de las 3 contraseñas anteriores**

Para aplicar dichas políticas de contraseña debemos modificar el módulo common-password de PAM añadiendo la siguiente línea:

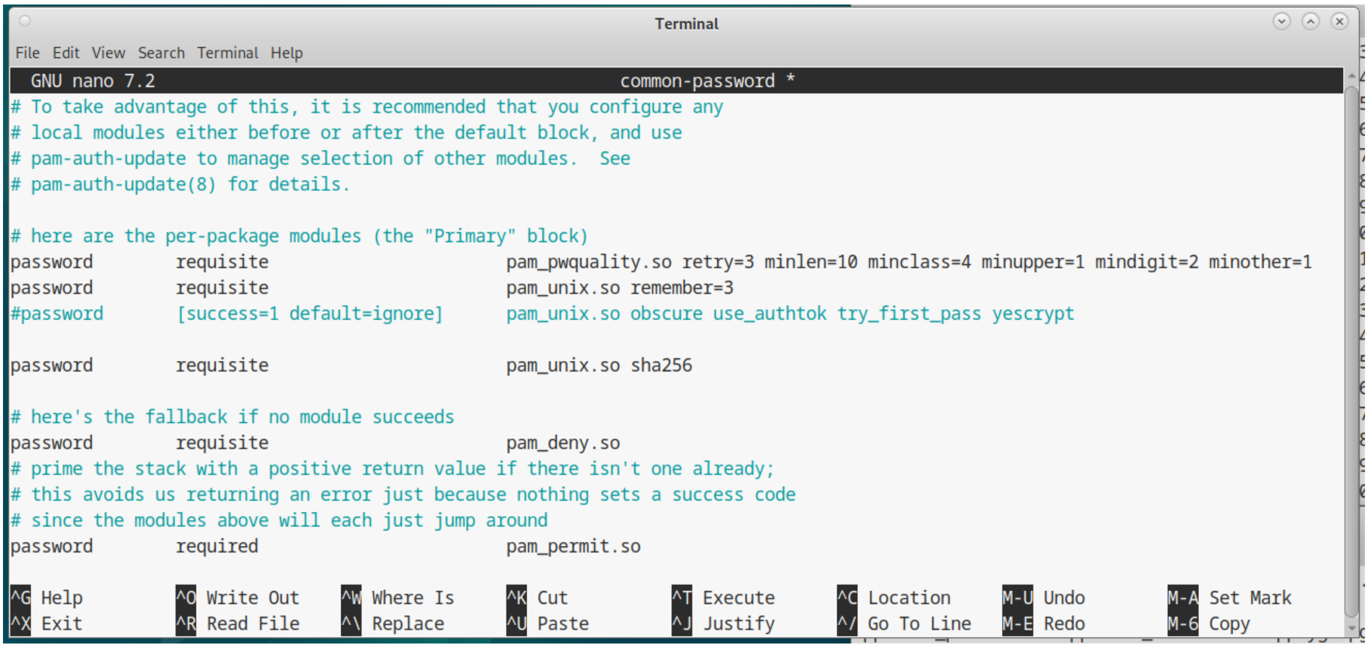
```
password requisite pam_pwquality.so retry=3 minlen=10 minclass=4 minupper=1
mindigit=2 minother=1
```



```
  GNU nano 7.2                                        common-password *
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
password        requisite                       pam_pwquality.so retry=3 minlen=10 minclass=4 minupper=1 mindigit=2 minother=1
#password       [success=1 default=ignore]      pam_unix.so obscure use_authtok try_first_pass yescrypt

password        requisite                       pam_unix.so sha256

# here's the fallback if no module succeeds
password        requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password        required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
```

Tras esto faltaría por establecer que no se puedan usar las 3 contraseñas anteriores, apara ello añadimos en el mismo fichero la siguiente línea:

```
password requisite pam_unix.so remember=3
```



```
  GNU nano 7.2                                        common-password *
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
password        requisite                       pam_pwquality.so retry=3 minlen=10 minclass=4 minupper=1 mindigit=2 minother=1
password        requisite                       pam_unix.so remember=3
#password       [success=1 default=ignore]      pam_unix.so obscure use_authtok try_first_pass yescrypt

password        requisite                       pam_unix.so sha256

# here's the fallback if no module succeeds
password        requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password        required                        pam_permit.so
```