Table of Contents

Chapter 27. introduction to users

This little chapter will teach you how to identify your user account on a Unix computer using commands like **who am i**, **id**, and more.

In a second part you will learn how to become another user with the **su** command.

And you will learn how to run a program as another user with **sudo**.

27.1. whoami

The **whoami** command tells you your username.

```
[paul@centos7 ~]$ whoami
paul
[paul@centos7 ~]$
```

27.2. who

The **who** command will give you information about who is logged on the system.

```
[paul@centos7 ~]$ who
root     pts/0        2014-10-10 23:07 (10.104.33.101)
paul     pts/1        2014-10-10 23:30 (10.104.33.101)
laura    pts/2        2014-10-10 23:34 (10.104.33.96)
tania    pts/3        2014-10-10 23:39 (10.104.33.91)
[paul@centos7 ~]$
```

27.3. who am i

With **who am i** the **who** command will display only the line pointing to your current session.

```
[paul@centos7 ~]$ who am i
paul     pts/1        2014-10-10 23:30 (10.104.33.101)
[paul@centos7 ~]$
```

27.4. w

The **w** command shows you who is logged on and what they are doing.

```
[paul@centos7 ~]$ w
 23:34:07 up 31 min,  2 users,  load average: 0.00, 0.01, 0.02
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0     23:07   15.00s  0.01s  0.01s top
paul     pts/1     23:30    7.00s  0.00s  0.00s w
[paul@centos7 ~]$
```

27.5. id

The **id** command will give you your user id, primary group id, and a list of the groups that you belong to.

```
paul@debian7:~$ id
uid=1000(paul) gid=1000(paul) groups=1000(paul)
```

On RHEL/CentOS you will also get **SELinux** context information with this command.

```
[root@centos7 ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r\
:unconfined_t:s0-s0:c0.c1023
```

27.6. su to another user

The **su** command allows a user to run a shell as another user.

```
laura@debian7:~$ su tania
Password:
tania@debian7:/home/laura$
```

27.7. su to root

Yes you can also **su** to become **root**, when you know the **root password**.

```
laura@debian7:~$ su root
Password:
root@debian7:/home/laura#
```

27.8. su as root

You need to know the password of the user you want to substitute to, unless your are logged in as **root**. The **root** user can become any existing user without knowing that user's password.

```
root@debian7:~# id
uid=0(root) gid=0(root) groups=0(root)
root@debian7:~# su - valentina
valentina@debian7:~$
```

27.9. su - $username

By default, the **su** command maintains the same shell environment. To become another user and also get the target user's environment, issue the **su -** command followed by the target username.

```
root@debian7:~# su laura
laura@debian7:/root$ exit
exit
root@debian7:~# su - laura
laura@debian7:~$ pwd
/home/laura
```

27.10. su -

When no username is provided to **su** or **su -**, the command will assume **root** is the target.

```
tania@debian7:~$ su -
Password:
root@debian7:~#
```

27.11. run a program as another user

The sudo program allows a user to start a program with the credentials of another user. Before this works, the system administrator has to set up the **/etc/sudoers** file. This can be useful to delegate administrative tasks to another user (without giving the root password).

The screenshot below shows the usage of **sudo**. User **paul** received the right to run **useradd** with the credentials of **root**. This allows **paul** to create new users on the system without becoming **root** and without knowing the **root password**.

First the command fails for **paul**.

```
paul@debian7:~$ /usr/sbin/useradd -m valentina
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

But with **sudo** it works.

```
paul@debian7:~$ sudo /usr/sbin/useradd -m valentina
[sudo] password for paul:
paul@debian7:~$
```

27.12. visudo

Check the man page of **visudo** before playing with the **/etc/sudoers** file. Editing the **sudoers** is out of scope for this fundamentals book.

```
paul@rhel65:~$ apropos visudo
visudo               (8)  - edit the sudoers file
paul@rhel65:~$
```

27.13. sudo su -

On some Linux systems like Ubuntu and Xubuntu, the **root** user does not have a password set. This means that it is not possible to login as **root** (extra security). To perform tasks as **root**, the first user is given all **sudo rights** via the **/etc/sudoers**. In fact all users that are members of the admin group can use sudo to run all commands as root.

```
root@laika:~# grep admin /etc/sudoers
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

The end result of this is that the user can type **sudo su -** and become root without having to enter the root password. The sudo command does require you to enter your own password. Thus the password prompt in the screenshot below is for sudo, not for su.

```
paul@laika:~$ sudo su -
Password:
root@laika:~#
```

27.14. sudo logging

Using **sudo** without authorization will result in a severe warning:

```
paul@rhel65:~$ sudo su -

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for paul:
paul is not in the sudoers file.  This incident will be reported.
paul@rhel65:~$
```

The root user can see this in the **/var/log/secure** on Red Hat and in **/var/log/auth.log** on Debian).

```
root@rhel65:~# tail /var/log/secure | grep sudo | tr -s ' '
Apr 13 16:03:42 rhel65 sudo: paul : user NOT in sudoers ; TTY=pts/0 ; PWD=\
/home/paul ; USER=root ; COMMAND=/bin/su -
root@rhel65:~#
```

27.15. practice: introduction to users

1. Run a command that displays only your currently logged on user name.

2. Display a list of all logged on users.

3. Display a list of all logged on users including the command they are running at this very moment.

4. Display your user name and your unique user identification (userid).

5. Use **su** to switch to another user account (unless you are root, you will need the password of the other account). And get back to the previous account.

6. Now use **su -** to switch to another user and notice the difference.

Note that **su -** gets you into the home directory of **Tania**.

7. Try to create a new user account (when using your normal user account). this should fail. (Details on adding user accounts are explained in the next chapter.)

8. Now try the same, but with **sudo** before your command.

27.16. solution: introduction to users

1. Run a command that displays only your currently logged on user name.

```
laura@debian7:~$ whoami
laura
laura@debian7:~$ echo $USER
laura
```

2. Display a list of all logged on users.

```
laura@debian7:~$ who
laura     pts/0        2014-10-13 07:22 (10.104.33.101)
laura@debian7:~$
```

3. Display a list of all logged on users including the command they are running at this very moment.

```
laura@debian7:~$ w
 07:47:02 up 16 min,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    10.104.33.101    07:30    6.00s  0.04s  0.00s w
root     pts/1    10.104.33.101    07:46    6.00s  0.01s  0.00s sleep 42
laura@debian7:~$
```

4. Display your user name and your unique user identification (userid).

```
laura@debian7:~$ id
uid=1005(laura) gid=1007(laura) groups=1007(laura)
laura@debian7:~$
```

5. Use **su** to switch to another user account (unless you are root, you will need the password of the other account). And get back to the previous account.

```
laura@debian7:~$ su tania
Password:
tania@debian7:/home/laura$ id
uid=1006(tania) gid=1008(tania) groups=1008(tania)
tania@debian7:/home/laura$ exit
laura@debian7:~$
```

6. Now use **su -** to switch to another user and notice the difference.

```
laura@debian7:~$ su - tania
Password:
tania@debian7:~$ pwd
/home/tania
tania@debian7:~$ logout
laura@debian7:~$
```

Note that **su -** gets you into the home directory of **Tania**.

7. Try to create a new user account (when using your normal user account). this should fail. (Details on adding user accounts are explained in the next chapter.)

```
laura@debian7:~$ useradd valentina
-su: useradd: command not found
laura@debian7:~$ /usr/sbin/useradd valentina
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
```

It is possible that **useradd** is located in **/sbin/useradd** on your computer.

8. Now try the same, but with **sudo** before your command.

```
laura@debian7:~$ sudo /usr/sbin/useradd valentina
[sudo] password for laura:
laura is not in the sudoers file.  This incident will be reported.
laura@debian7:~$
```

Notice that **laura** has no permission to use the **sudo** on this system.

Chapter 28. user management

This chapter will teach you how to use **useradd**, **usermod** and **userdel** to create, modify and remove user accounts.

You will need **root** access on a Linux computer to complete this chapter.

28.1. user management

User management on Linux can be done in three complementary ways. You can use the **graphical** tools provided by your distribution. These tools have a look and feel that depends on the distribution. If you are a novice Linux user on your home system, then use the graphical tool that is provided by your distribution. This will make sure that you do not run into problems.

Another option is to use **command line tools** like useradd, usermod, gpasswd, passwd and others. Server administrators are likely to use these tools, since they are familiar and very similar across many different distributions. This chapter will focus on these command line tools.

A third and rather extremist way is to **edit the local configuration files** directly using vi (or vipw/vigr). Do not attempt this as a novice on production systems!

28.2. /etc/passwd

The local user database on Linux (and on most Unixes) is **/etc/passwd**.

```
[root@RHEL5 ~]# tail /etc/passwd
inge:x:518:524:art dealer:/home/inge:/bin/ksh
ann:x:519:525:flute player:/home/ann:/bin/bash
frederik:x:520:526:rubius poet:/home/frederik:/bin/bash
steven:x:521:527:roman emperor:/home/steven:/bin/bash
pascale:x:522:528:artist:/home/pascale:/bin/ksh
geert:x:524:530:kernel developer:/home/geert:/bin/bash
wim:x:525:531:master damuti:/home/wim:/bin/bash
sandra:x:526:532:radish stresser:/home/sandra:/bin/bash
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

As you can see, this file contains seven columns separated by a colon. The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory, and the login shell.

More information can be found by typing **man 5 passwd**.

```
[root@RHEL5 ~]# man 5 passwd
```

28.3. root

The **root** user also called the **superuser** is the most powerful account on your Linux system. This user can do almost anything, including the creation of other users. The root user always has userid 0 (regardless of the name of the account).

```
[root@RHEL5 ~]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

28.4. useradd

You can add users with the **useradd** command. The example below shows how to add a user named yanina (last parameter) and at the same time forcing the creation of the home directory (-m), setting the name of the home directory (-d), and setting a description (-c).

```
[root@RHEL5 ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina
[root@RHEL5 ~]# tail -1 /etc/passwd
yanina:x:529:529:yanina wickmayer:/home/yanina:/bin/bash
```

The user named yanina received userid 529 and **primary group** id 529.

28.5. /etc/default/useradd

Both Red Hat Enterprise Linux and Debian/Ubuntu have a file called **/etc/default/useradd** that contains some default user options. Besides using cat to display this file, you can also use **useradd -D**.

```
[root@RHEL4 ~]# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

28.6. userdel

You can delete the user yanina with **userdel**. The -r option of userdel will also remove the home directory.

```
[root@RHEL5 ~]# userdel -r yanina
```

28.7. usermod

You can modify the properties of a user with the **usermod** command. This example uses **usermod** to change the description of the user harry.

```
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:harry potter:/home/harry:/bin/bash
[root@RHEL4 ~]# usermod -c 'wizard' harry
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:wizard:/home/harry:/bin/bash
```

28.8. creating home directories

The easiest way to create a home directory is to supply the **-m** option with **useradd** (it is likely set as a default option on Linux).

A less easy way is to create a home directory manually with **mkdir** which also requires setting the owner and the permissions on the directory with **chmod** and **chown** (both commands are discussed in detail in another chapter).

```
[root@RHEL5 ~]# mkdir /home/laura
[root@RHEL5 ~]# chown laura:laura /home/laura
[root@RHEL5 ~]# chmod 700 /home/laura
[root@RHEL5 ~]# ls -ld /home/laura/
drwx------ 2 laura laura 4096 Jun 24 15:17 /home/laura/
```

28.9. /etc/skel/

When using **useradd** the **-m** option, the **/etc/skel/** directory is copied to the newly created home directory. The **/etc/skel/** directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way **/etc/skel/** serves as a default home directory and as a default user profile.

```
[root@RHEL5 ~]# ls -la /etc/skel/
total 48
drwxr-xr-x  2 root root  4096 Apr  1 00:11 .
drwxr-xr-x 97 root root 12288 Jun 24 15:36 ..
-rw-r--r--  1 root root    24 Jul 12  2006 .bash_logout
-rw-r--r--  1 root root   176 Jul 12  2006 .bash_profile
-rw-r--r--  1 root root   124 Jul 12  2006 .bashrc
```

28.10. deleting home directories

The -r option of **userdel** will make sure that the home directory is deleted together with the user account.

```
[root@RHEL5 ~]# ls -ld /home/wim/
drwx------ 2 wim wim 4096 Jun 24 15:19 /home/wim/
[root@RHEL5 ~]# userdel -r wim
[root@RHEL5 ~]# ls -ld /home/wim/
ls: /home/wim/: No such file or directory
```

28.11. login shell

The **/etc/passwd** file specifies the **login shell** for the user. In the screenshot below you can see that user annelies will log in with the **/bin/bash** shell, and user laura with the **/bin/ksh** shell.

```
[root@RHEL5 ~]# tail -2 /etc/passwd
annelies:x:527:533:sword fighter:/home/annelies:/bin/bash
laura:x:528:534:art dealer:/home/laura:/bin/ksh
```

You can use the usermod command to change the shell for a user.

```
[root@RHEL5 ~]# usermod -s /bin/bash laura
[root@RHEL5 ~]# tail -1 /etc/passwd
laura:x:528:534:art dealer:/home/laura:/bin/bash
```

28.12. chsh

Users can change their login shell with the **chsh** command. First, user harry obtains a list of available shells (he could also have done a **cat /etc/shells**) and then changes his login shell to the **Korn shell** (/bin/ksh). At the next login, harry will default into ksh instead of bash.

```
[laura@centos7 ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
/bin/ksh
/bin/tcsh
/bin/csh
[laura@centos7 ~]$
```

Note that the **-l** option does not exist on Debian and that the above screenshot assumes that **ksh** and **csh** shells are installed.

The screenshot below shows how **laura** can change her default shell (active on next login).

```
[laura@centos7 ~]$ chsh -s /bin/ksh
Changing shell for laura.
Password:
Shell changed.
```

28.13. practice: user management

1. Create a user account named **serena**, including a home directory and a description (or comment) that reads **Serena Williams**. Do all this in one single command.

2. Create a user named **venus**, including home directory, bash shell, a description that reads **Venus Williams** all in one single command.

3. Verify that both users have correct entries in **/etc/passwd**, **/etc/shadow** and **/etc/group**.

4. Verify that their home directory was created.

5. Create a user named **einstime** with **/bin/date** as his default logon shell.

7. What happens when you log on with the **einstime** user ? Can you think of a useful real world example for changing a user's login shell to an application ?

8. Create a file named **welcome.txt** and make sure every new user will see this file in their home directory.

9. Verify this setup by creating (and deleting) a test user account.

10. Change the default login shell for the **serena** user to **/bin/bash**. Verify before and after you make this change.

28.14. solution: user management

1. Create a user account named **serena**, including a home directory and a description (or comment) that reads **Serena Williams**. Do all this in one single command.

```
root@debian7:~# useradd -m -c 'Serena Williams' serena
```

2. Create a user named **venus**, including home directory, bash shell, a description that reads **Venus Williams** all in one single command.

```
root@debian7:~# useradd -m -c "Venus Williams" -s /bin/bash venus
```

3. Verify that both users have correct entries in **/etc/passwd**, **/etc/shadow** and **/etc/group**.

```
root@debian7:~# tail -2 /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/sh
venus:x:1009:1011:Venus Williams:/home/venus:/bin/bash
root@debian7:~# tail -2 /etc/shadow
serena:!:16358:0:99999:7:::
venus:!:16358:0:99999:7:::
root@debian7:~# tail -2 /etc/group
serena:x:1010:
venus:x:1011:
```

4. Verify that their home directory was created.

```
root@debian7:~# ls -lrt /home | tail -2
drwxr-xr-x 2 serena    serena    4096 Oct 15 10:50 serena
drwxr-xr-x 2 venus     venus     4096 Oct 15 10:59 venus
root@debian7:~#
```

5. Create a user named **einstime** with **/bin/date** as his default logon shell.

```
root@debian7:~# useradd -s /bin/date einstime
```

Or even better:

```
root@debian7:~# useradd -s $(which date) einstime
```

7. What happens when you log on with the **einstime** user ? Can you think of a useful real world example for changing a user's login shell to an application ?

```
root@debian7:~# su - einstime
Wed Oct 15 11:05:56 UTC 2014 # You get the output of the date command
root@debian7:~#
```

It can be useful when users need to access only one application on the server. Just logging in opens the application for them, and closing the application automatically logs them out.

8. Create a file named **welcome.txt** and make sure every new user will see this file in their home directory.

```
root@debian7:~# echo Hello > /etc/skel/welcome.txt
```

9. Verify this setup by creating (and deleting) a test user account.

```
root@debian7:~# useradd -m test
root@debian7:~# ls -l /home/test
total 4
-rw-r--r-- 1 test test 6 Oct 15 11:16 welcome.txt
root@debian7:~# userdel -r test
root@debian7:~#
```

10. Change the default login shell for the **serena** user to **/bin/bash**. Verify before and after you make this change.

```
root@debian7:~# grep serena /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/sh
root@debian7:~# usermod -s /bin/bash serena
root@debian7:~# grep serena /etc/passwd
serena:x:1008:1010:Serena Williams:/home/serena:/bin/bash
root@debian7:~#
```

Chapter 29. user passwords

This chapter will tell you more about passwords for local users.

Three methods for setting passwords are explained; using the **passwd** command, using **openssel passwd**, and using the **crypt** function in a C program.

The chapter will also discuss password settings and disabling, suspending or locking accounts.

29.1. passwd

Passwords of users can be set with the **passwd** command. Users will have to provide their old password before twice entering the new one.

```
[tania@centos7 ~]$ passwd
Changing password for user tania.
Changing password for tania.
(current) UNIX password:
New password:
BAD PASSWORD: The password is shorter than 8 characters
New password:
BAD PASSWORD: The password is a palindrome
New password:
BAD PASSWORD: The password is too similar to the old one
passwd: Have exhausted maximum number of retries for service
```

As you can see, the passwd tool will do some basic verification to prevent users from using too simple passwords. The **root** user does not have to follow these rules (there will be a warning though). The **root** user also does not have to provide the old password before entering the new password twice.

```
root@debian7:~# passwd tania
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

29.2. shadow file

User passwords are encrypted and kept in **/etc/shadow**. The /etc/shadow file is read only and can only be read by root. We will see in the file permissions section how it is possible for users to change their password. For now, you will have to know that users can change their password with the **/usr/bin/passwd** command.

```
[root@centos7 ~]# tail -4 /etc/shadow
paul:$6$ikp2Xta5BT.Tml.p$2TZjNnOYNNQKpwLJqoGJbVsZG5/Fti8ovBRd.VzRbiDSl7TEq\
IaSMH.TeBKnTS/SjlMruW8qffC0JNORW.BTW1:16338:0:99999:7:::
tania:$6$8Z/zovxj$9qvoqT8i9KIrmN.k4EQwAF5ryz5yzNwEvYjAa9L5XVXQu.z4DlpvMREH\
eQpQzvRnqFdKkVj17H5ST.c79HDZw0:16356:0:99999:7:::
laura:$6$glDuTY5e$/NYYWLxfHgZFWeoujaXSMcR.Mz.lGOxtcxFocFVJNb98nbTPhWFXfKWG\
SyYh1WCv6763Wq54.w24Yr3uAZBOm/:16356:0:99999:7:::
valentina:$6$jrZa6PVI$1uQgqR6En9mZB6mKJ3LXRB4CnFko6LRhbh.v4iqUk9MVreui1lv7\
GxHOUDSKA0N55ZRNhGHa6T2ouFnVno/0o1:16356:0:99999:7:::
[root@centos7 ~]#
```

The **/etc/shadow** file contains nine colon separated columns. The nine fields contain (from left to right) the user name, the encrypted password (note that only inge and laura have an encrypted password), the day the password was last changed (day 1 is January 1, 1970), number of days the password must be left unchanged, password expiry day, warning number of days before password expiry, number of days after expiry before disabling the account, and the day the account was disabled (again, since 1970). The last field has no meaning yet.

All the passwords in the screenshot above are hashes of **hunter2**.

29.3. encryption with passwd

Passwords are stored in an encrypted format. This encryption is done by the **crypt** function. The easiest (and recommended) way to add a user with a password to the system is to add the user with the **useradd -m user** command, and then set the user's password with **passwd**.

```
[root@RHEL4 ~]# useradd -m xavier
[root@RHEL4 ~]# passwd xavier
Changing password for user xavier.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@RHEL4 ~]#
```

29.4. encryption with openssl

Another way to create users with a password is to use the -p option of useradd, but that option requires an encrypted password. You can generate this encrypted password with the **openssl passwd** command.

The **openssl passwd** command will generate several distinct hashes for the same password, for this it uses a **salt**.

```
paul@rhel65:~$ openssl passwd hunter2
86jcUNlnGDFpY
paul@rhel65:~$ openssl passwd hunter2
Yj7mDO9OAnvq6
paul@rhel65:~$ openssl passwd hunter2
YqDcJeGoDbzKA
paul@rhel65:~$
```

This **salt** can be chosen and is visible as the first two characters of the hash.

```
paul@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
paul@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
paul@rhel65:~$ openssl passwd -salt 42 hunter2
42ZrbtP1Ze8G.
paul@rhel65:~$
```

This example shows how to create a user with password.

```
root@rhel65:~# useradd -m -p $(openssl passwd hunter2) mohamed
```

Note that this command puts the password in your command history!

29.5. encryption with crypt

A third option is to create your own C program using the crypt function, and compile this into a command.

```
paul@rhel65:~$ cat MyCrypt.c
#include <stdio.h>
#define __USE_XOPEN
#include <unistd.h>

int main(int argc, char** argv)
{
 if(argc==3)
   {
       printf("%s\n", crypt(argv[1],argv[2]));
   }
   else
   {
       printf("Usage: MyCrypt $password $salt\n" );
   }
  return 0;
}
```

This little program can be compiled with **gcc** like this.

```
paul@rhel65:~$ gcc MyCrypt.c -o MyCrypt -lcrypt
```

To use it, we need to give two parameters to MyCrypt. The first is the unencrypted password, the second is the salt. The salt is used to perturb the encryption algorithm in one of 4096 different ways. This variation prevents two users with the same password from having the same entry in **/etc/shadow**.

```
paul@rhel65:~$ ./MyCrypt hunter2 42
42ZrbtP1Ze8G.
paul@rhel65:~$ ./MyCrypt hunter2 33
33d6taYSiEUXI
```

Did you notice that the first two characters of the password are the **salt**?

The standard output of the crypt function is using the DES algorithm which is old and can be cracked in minutes. A better method is to use **md5** passwords which can be recognized by a salt starting with 1.

```
paul@rhel65:~$ ./MyCrypt hunter2 '$1$42'
$1$42$7l6Y3xT5282XmZrtDOF9f0
paul@rhel65:~$ ./MyCrypt hunter2 '$6$42'
$6$42$OqFFAVnI3gTSYG0yI9TZWX9cpyQzwIop7HwpG1LLEsNBiMr4w6OvLX1KDa./UpwXfrFk1i...
```

The **md5** salt can be up to eight characters long. The salt is displayed in **/etc/shadow** between the second and third $, so never use the password as the salt!

```
paul@rhel65:~$ ./MyCrypt hunter2 '$1$hunter2'
$1$hunter2$YVxrxDmidq7Xf8Gdt6qM2.
```

29.6. /etc/login.defs

The **/etc/login.defs** file contains some default settings for user passwords like password aging and length settings. (You will also find the numerical limits of user ids and group ids and whether or not a home directory should be created by default).

```
root@rhel65:~# grep ^PASS /etc/login.defs
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

Debian also has this file.

```
root@debian7:~# grep PASS /etc/login.defs
#  PASS_MAX_DAYS   Maximum number of days a password may be used.
#  PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#  PASS_WARN_AGE   Number of days warning given before a password expires.
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
#PASS_CHANGE_TRIES
#PASS_ALWAYS_WARN
#PASS_MIN_LEN
#PASS_MAX_LEN
# NO_PASSWORD_CONSOLE
root@debian7:~#
```

29.7. chage

The **chage** command can be used to set an expiration date for a user account (-E), set a minimum (-m) and maximum (-M) password age, a password expiration date, and set the number of warning days before the password expiration date. Much of this functionality is also available from the **passwd** command. The **-l** option of chage will list these settings for a user.

```
root@rhel65:~# chage -l paul
Last password change                                    : Mar 27, 2014
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
root@rhel65:~#
```

29.8. disabling a password

Passwords in **/etc/shadow** cannot begin with an exclamation mark. When the second field in **/etc/passwd** starts with an exclamation mark, then the password can not be used.

Using this feature is often called **locking**, **disabling**, or **suspending** a user account. Besides **vi** (or vipw) you can also accomplish this with **usermod**.

The first command in the next screenshot will show the hashed password of **laura** in **/etc/shadow**. The next command disables the password of **laura**, making it impossible for Laura to authenticate using this password.

```
root@debian7:~# grep laura /etc/shadow | cut -c1-70
laura:$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJV
root@debian7:~# usermod -L laura
```

As you can see below, the password hash is simply preceded with an exclamation mark.

```
root@debian7:~# grep laura /etc/shadow | cut -c1-70
laura:!$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJ
root@debian7:~#
```

The root user (and users with **sudo** rights on **su**) still will be able to **su** into the **laura** account (because the password is not needed here). Also note that **laura** will still be able to login if she has set up passwordless ssh!

```
root@debian7:~# su - laura
laura@debian7:~$
```

You can unlock the account again with **usermod -U**.

```
root@debian7:~# usermod -U laura
root@debian7:~# grep laura /etc/shadow | cut -c1-70
laura:$6$JYj4JZqp$stwwWACp3OtE1R2aZuE87j.nbW.puDkNUYVk7mCHfCVMa3CoDUJV
```

Watch out for tiny differences in the command line options of **passwd**, **usermod**, and **useradd** on different Linux distributions. Verify the local files when using features like **"disabling, suspending, or locking"** on user accounts and their passwords.

29.9. editing local files

If you still want to manually edit the **/etc/passwd** or **/etc/shadow**, after knowing these commands for password management, then use **vipw** instead of vi(m) directly. The **vipw** tool will do proper locking of the file.

```
[root@RHEL5 ~]# vipw /etc/passwd
vipw: the password file is busy (/etc/ptmp present)
```

29.10. practice: user passwords

1. Set the password for **serena** to **hunter2**.

2. Also set a password for **venus** and then lock the **venus** user account with **usermod**. Verify the locking in **/etc/shadow** before and after you lock it.

3. Use **passwd -d** to disable the **serena** password. Verify the **serena** line in **/etc/shadow** before and after disabling.

4. What is the difference between locking a user account and disabling a user account's password like we just did with **usermod -L** and **passwd -d**?

5. Try changing the password of serena to serena as serena.

6. Make sure **serena** has to change her password in 10 days.

7. Make sure every new user needs to change their password every 10 days.

8. Take a backup as root of **/etc/shadow**. Use **vi** to copy an encrypted **hunter2** hash from **venus** to **serena**. Can **serena** now log on with **hunter2** as a password ?

9. Why use **vipw** instead of **vi** ? What could be the problem when using **vi** or **vim** ?

10. Use **chsh** to list all shells (only works on RHEL/CentOS/Fedora), and compare to **cat /etc/shells**.

11. Which **useradd** option allows you to name a home directory ?

12. How can you see whether the password of user **serena** is locked or unlocked ? Give a solution with **grep** and a solution with **passwd**.

29.11. solution: user passwords

1. Set the password for **serena** to **hunter2**.

```
root@debian7:~# passwd serena
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

2. Also set a password for **venus** and then lock the **venus** user account with **usermod**. Verify the locking in **/etc/shadow** before and after you lock it.

```
root@debian7:~# passwd venus
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@debian7:~# grep venus /etc/shadow | cut -c1-70
venus:$6$gswzXICW$uSnKFV1kFKZmTPaMVS4AvNA/KO27OxN0v5LHdV9ed0gTyXrjUeM/
root@debian7:~# usermod -L venus
root@debian7:~# grep venus /etc/shadow | cut -c1-70
venus:!$6$gswzXICW$uSnKFV1kFKZmTPaMVS4AvNA/KO27OxN0v5LHdV9ed0gTyXrjUeM
```

Note that **usermod -L** precedes the password hash with an exclamation mark (!).

3. Use **passwd -d** to disable the **serena** password. Verify the **serena** line in **/etc/shadow** before and after disabling.

```
root@debian7:~# grep serena /etc/shadow | cut -c1-70
serena:$6$Es/omrPE$F2Ypu8kpLrfKdW0v/UIwA5jrYyBD2nwZ/dt.i/IypRgiPZSdB/B
root@debian7:~# passwd -d serena
passwd: password expiry information changed.
root@debian7:~# grep serena /etc/shadow
serena::16358:0:99999:7:::
root@debian7:~#
```

4. What is the difference between locking a user account and disabling a user account's password like we just did with **usermod -L** and **passwd -d**?

Locking will prevent the user from logging on to the system with his password by putting a ! in front of the password in **/etc/shadow**.

Disabling with **passwd** will erase the password from **/etc/shadow**.

5. Try changing the password of serena to serena as serena.

```
log on as serena, then execute: passwd serena... it should fail!
```

6. Make sure **serena** has to change her password in 10 days.

```
chage -M 10 serena
```

7. Make sure every new user needs to change their password every 10 days.

```
vi /etc/login.defs (and change PASS_MAX_DAYS to 10)
```

8. Take a backup as root of **/etc/shadow**. Use **vi** to copy an encrypted **hunter2** hash from **venus** to **serena**. Can **serena** now log on with **hunter2** as a password ?

```
Yes.
```

9. Why use **vipw** instead of **vi** ? What could be the problem when using **vi** or **vim** ?

```
vipw will give a warning when someone else is already using that file (with vipw).
```

10. Use **chsh** to list all shells (only works on RHEL/CentOS/Fedora), and compare to **cat /etc/shells**.

```
chsh -l
cat /etc/shells
```

11. Which **useradd** option allows you to name a home directory ?

```
-d
```

12. How can you see whether the password of user **serena** is locked or unlocked ? Give a solution with **grep** and a solution with **passwd**.

```
grep serena /etc/shadow
```

```
passwd -S serena
```

Chapter 30. user profiles

Logged on users have a number of preset (and customized) aliases, variables, and functions, but where do they come from ? The **shell** uses a number of startup files that are executed (or rather **sourced**) whenever the shell is invoked. What follows is an overview of startup scripts.

30.1. system profile

Both the **bash** and the **ksh** shell will verify the existence of **/etc/profile** and **source** it if it exists.

When reading this script, you will notice (both on Debian and on Red Hat Enterprise Linux) that it builds the PATH environment variable (among others). The script might also change the PS1 variable, set the HOSTNAME and execute even more scripts like **/etc/inputrc**

This screenshot uses grep to show PATH manipulation in **/etc/profile** on Debian.

```
root@debian7:~# grep PATH /etc/profile
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
export PATH
root@debian7:~#
```

This screenshot uses grep to show PATH manipulation in **/etc/profile** on RHEL7/CentOS7.

```
[root@centos7 ~]# grep PATH /etc/profile
    case ":${PATH}:" in
                PATH=$PATH:$1
                PATH=$1:$PATH
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
[root@centos7 ~]#
```

The **root user** can use this script to set aliases, functions, and variables for every user on the system.

30.2. ~/.bash_profile

When this file exists in the home directory, then **bash** will source it. On Debian Linux 5/6/7 this file does not exist by default.

RHEL7/CentOS7 uses a small **~/.bash_profile** where it checks for the existence of **~/.bashrc** and then sources it. It also adds $HOME/bin to the $PATH variable.

```
[root@rhel7 ~]# cat /home/paul/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH
[root@rhel7 ~]#
```

30.3. ~/.bash_login

When **.bash_profile** does not exist, then **bash** will check for **~/.bash_login** and source it.

Neither Debian nor Red Hat have this file by default.

30.4. ~/.profile

When neither **~/.bash_profile** and **~/.bash_login** exist, then bash will verify the existence of **~/.profile** and execute it. This file does not exist by default on Red Hat.

On Debian this script can execute **~/.bashrc** and will add $HOME/bin to the $PATH variable.

```
root@debian7:~# tail -11 /home/paul/.profile
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

RHEL/CentOS does not have this file by default.

30.5. ~/.bashrc

The **~/.bashrc** script is often sourced by other scripts. Let us take a look at what it does by default.

Red Hat uses a very simple **~/.bashrc**, checking for **/etc/bashrc** and sourcing it. It also leaves room for custom aliases and functions.

```
[root@rhel7 ~]# cat /home/paul/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

On Debian this script is quite a bit longer and configures $PS1, some history variables and a number af active and inactive aliases.

```
root@debian7:~# wc -l /home/paul/.bashrc
110 /home/paul/.bashrc
```

30.6. ~/.bash_logout

When exiting **bash**, it can execute **~/.bash_logout**.

Debian use this opportunity to clear the console screen.

```
serena@deb503:~$ cat .bash_logout
# ~/.bash_logout: executed by bash(1) when login shell exits.

# when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
```

Red Hat Enterprise Linux 5 will simple call the **/usr/bin/clear** command in this script.

```
[serena@rhel53 ~]$ cat .bash_logout
# ~/.bash_logout

/usr/bin/clear
```

Red Hat Enterprise Linux 6 and 7 create this file, but leave it empty (except for a comment).

```
paul@rhel65:~$ cat .bash_logout
# ~/.bash_logout
```

30.7. Debian overview

Below is a table overview of when Debian is running any of these bash startup scripts.

Table 30.1. Debian User Environment

script	su	su -	ssh	gdm
~./bashrc	no	yes	yes	yes
~/.profile	no	yes	yes	yes
/etc/profile	no	yes	yes	yes
/etc/bash.bashrc	yes	no	no	yes

30.8. RHEL5 overview

Below is a table overview of when Red Hat Enterprise Linux 5 is running any of these bash startup scripts.

Table 30.2. Red Hat User Environment

script	su	su -	ssh	gdm
~./bashrc	yes	yes	yes	yes
~/.bash_profile	no	yes	yes	yes
/etc/profile	no	yes	yes	yes
/etc/bashrc	yes	yes	yes	yes

30.9. practice: user profiles

1. Make a list of all the profile files on your system.

2. Read the contents of each of these, often they **source** extra scripts.

3. Put a unique variable, alias and function in each of those files.

4. Try several different ways to obtain a shell (su, su -, ssh, tmux, gnome-terminal, Ctrl-alt-F1, ...) and verify which of your custom variables, aliases and function are present in your environment.

5. Do you also know the order in which they are executed?

6. When an application depends on a setting in $HOME/.profile, does it matter whether $HOME/.bash_profile exists or not ?

30.10. solution: user profiles

1. Make a list of all the profile files on your system.

```
ls -a ~ ; ls -l /etc/pro* /etc/bash*
```

2. Read the contents of each of these, often they **source** extra scripts.

3. Put a unique variable, alias and function in each of those files.

4. Try several different ways to obtain a shell (su, su -, ssh, tmux, gnome-terminal, Ctrl-alt-F1, ...) and verify which of your custom variables, aliases and function are present in your environment.

5. Do you also know the order in which they are executed?

```
same name aliases, functions and variables will overwrite each other
```

6. When an application depends on a setting in $HOME/.profile, does it matter whether $HOME/.bash_profile exists or not ?

```
Yes it does matter. (man bash /INVOCATION)
```

Chapter 31. groups

Users can be listed in **groups**. Groups allow you to set permissions on the group level instead of having to set permissions for every individual user.

Every Unix or Linux distribution will have a graphical tool to manage groups. Novice users are advised to use this graphical tool. More experienced users can use command line tools to manage users, but be careful: Some distributions do not allow the mixed use of GUI and CLI tools to manage groups (YaST in Novell Suse). Senior administrators can edit the relevant files directly with **vi** or **vigr**.

31.1. groupadd

Groups can be created with the **groupadd** command. The example below shows the creation of five (empty) groups.

```
root@laika:~# groupadd tennis
root@laika:~# groupadd football
root@laika:~# groupadd snooker
root@laika:~# groupadd formula1
root@laika:~# groupadd salsa
```

31.2. group file

Users can be a member of several groups. Group membership is defined by the **/etc/group** file.

```
root@laika:~# tail -5 /etc/group
tennis:x:1006:
football:x:1007:
snooker:x:1008:
formula1:x:1009:
salsa:x:1010:
root@laika:~#
```

The first field is the group's name. The second field is the group's (encrypted) password (can be empty). The third field is the group identification or **GID**. The fourth field is the list of members, these groups have no members.

31.3. groups

A user can type the **groups** command to see a list of groups where the user belongs to.

```
[harry@RHEL4b ~]$ groups
harry sports
[harry@RHEL4b ~]$
```

31.4. usermod

Group membership can be modified with the useradd or **usermod** command.

```
root@laika:~# usermod -a -G tennis inge
root@laika:~# usermod -a -G tennis katrien
root@laika:~# usermod -a -G salsa katrien
root@laika:~# usermod -a -G snooker sandra
root@laika:~# usermod -a -G formula1 annelies
root@laika:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
snooker:x:1008:sandra
formula1:x:1009:annelies
salsa:x:1010:katrien
root@laika:~#
```

Be careful when using **usermod** to add users to groups. By default, the **usermod** command will **remove** the user from every group of which he is a member if the group is not listed in the command! Using the **-a** (append) switch prevents this behaviour.

31.5. groupmod

You can change the group name with the **groupmod** command.

```
root@laika:~# groupmod -n darts snooker
root@laika:~# tail -5 /etc/group
tennis:x:1006:inge,katrien
football:x:1007:
formula1:x:1009:annelies
salsa:x:1010:katrien
darts:x:1008:sandra
```

31.6. groupdel

You can permanently remove a group with the **groupdel** command.

```
root@laika:~# groupdel tennis
root@laika:~#
```

31.7. gpasswd

You can delegate control of group membership to another user with the **gpasswd** command. In the example below we delegate permissions to add and remove group members to serena for the sports group. Then we **su** to serena and add harry to the sports group.

```
[root@RHEL4b ~]# gpasswd -A serena sports
[root@RHEL4b ~]# su - serena
[serena@RHEL4b ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry)
[serena@RHEL4b ~]$ gpasswd -a harry sports
Adding user harry to group sports
[serena@RHEL4b ~]$ id harry
uid=516(harry) gid=520(harry) groups=520(harry),522(sports)
[serena@RHEL4b ~]$ tail -1 /etc/group
sports:x:522:serena,venus,harry
[serena@RHEL4b ~]$
```

Group administrators do not have to be a member of the group. They can remove themselves from a group, but this does not influence their ability to add or remove members.

```
[serena@RHEL4b ~]$ gpasswd -d serena sports
Removing user serena from group sports
[serena@RHEL4b ~]$ exit
```

Information about group administrators is kept in the **/etc/gshadow** file.

```
[root@RHEL4b ~]# tail -1 /etc/gshadow
sports:!:serena:venus,harry
[root@RHEL4b ~]#
```

To remove all group administrators from a group, use the **gpasswd** command to set an empty administrators list.

```
[root@RHEL4b ~]# gpasswd -A "" sports
```

31.8. newgrp

You can start a **child shell** with a new temporary **primary group** using the **newgrp** command.

```
root@rhel65:~# mkdir prigroup
root@rhel65:~# cd prigroup/
root@rhel65:~/prigroup# touch standard.txt
root@rhel65:~/prigroup# ls -l
total 0
-rw-r--r--. 1 root root 0 Apr 13 17:49 standard.txt
root@rhel65:~/prigroup# echo $SHLVL
1
root@rhel65:~/prigroup# newgrp tennis
root@rhel65:~/prigroup# echo $SHLVL
2
root@rhel65:~/prigroup# touch newgrp.txt
root@rhel65:~/prigroup# ls -l
total 0
-rw-r--r--. 1 root tennis 0 Apr 13 17:49 newgrp.txt
-rw-r--r--. 1 root root   0 Apr 13 17:49 standard.txt
root@rhel65:~/prigroup# exit
exit
root@rhel65:~/prigroup#
```

31.9. vigr

Similar to vipw, the **vigr** command can be used to manually edit the **/etc/group** file, since it will do proper locking of the file. Only experienced senior administrators should use **vi** or **vigr** to manage groups.