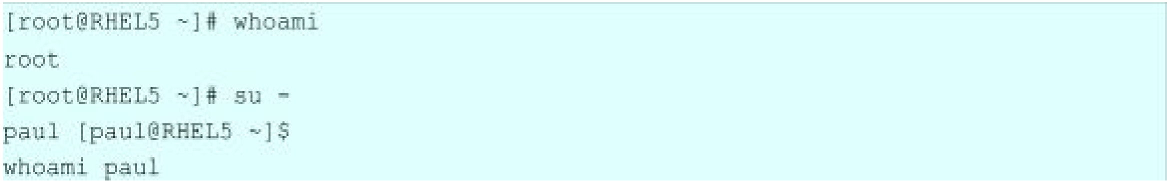
Chapter 25. users

25.1. identify yourself

25.1.1. whoami

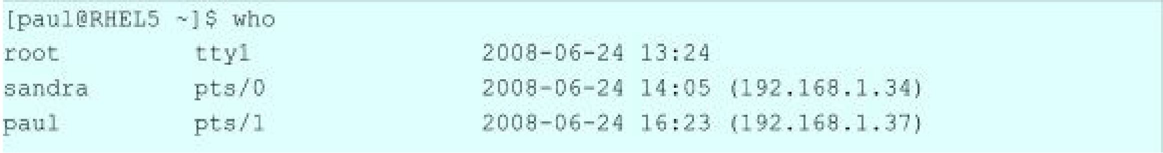
The **whoami** command tells you your username.



```
[root@RHEL5 ~]# whoami
root
[root@RHEL5 ~]# su -
paul [paul@RHEL5 ~]$
whoami paul
```

25.1.2. who

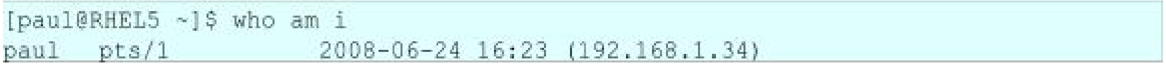
The **who** command will give you information about who is logged on the system.



```
[paul@RHEL5 ~]$ who
root          tty1               2008-06-24 13:24
sandra        pts/0              2008-06-24 14:05 (192.168.1.34)
paul          pts/1              2008-06-24 16:23 (192.168.1.37)
```

25.1.3. who am i

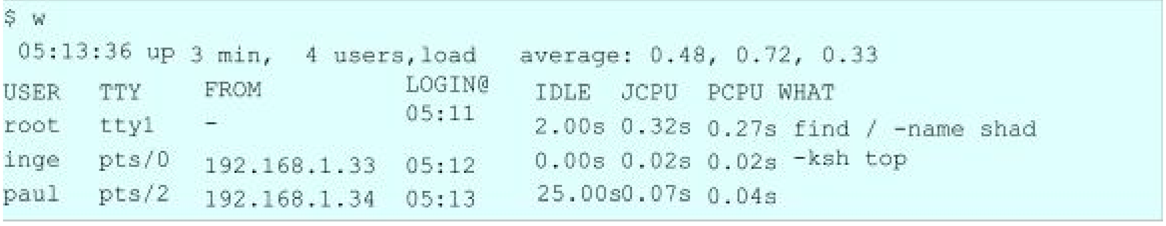
With **who am i** the who command will display only the line pointing to your current session.



```
[paul@RHEL5 ~]$ who am i
paul   pts/1          2008-06-24 16:23 (192.168.1.34)
```

25.1.4. w

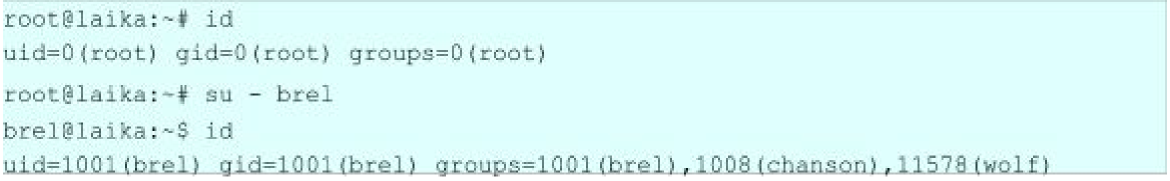
The **w** command shows you who is logged on and what they are doing.



```
$ w
 05:13:36 up 3 min,  4 users,load   average: 0.48, 0.72, 0.33
USER   TTY    FROM           LOGIN@    IDLE   JCPU   PCPU WHAT
root   tty1   -              05:11     2.00s  0.32s  0.27s find / -name shad
inge   pts/0  192.168.1.33   05:12     0.00s  0.02s  0.02s -ksh top
paul   pts/2  192.168.1.34   05:13     25.00s0.07s   0.04s
```

25.1.5. id

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



```
root@laika:~# id
uid=0(root) gid=0(root) groups=0(root)
root@laika:~# su - brel
brel@laika:~$ id
uid=1001(brel) gid=1001(brel) groups=1001(brel),1008(chanson),11578(wolf)
```

25.2. users

25.2.1. user management

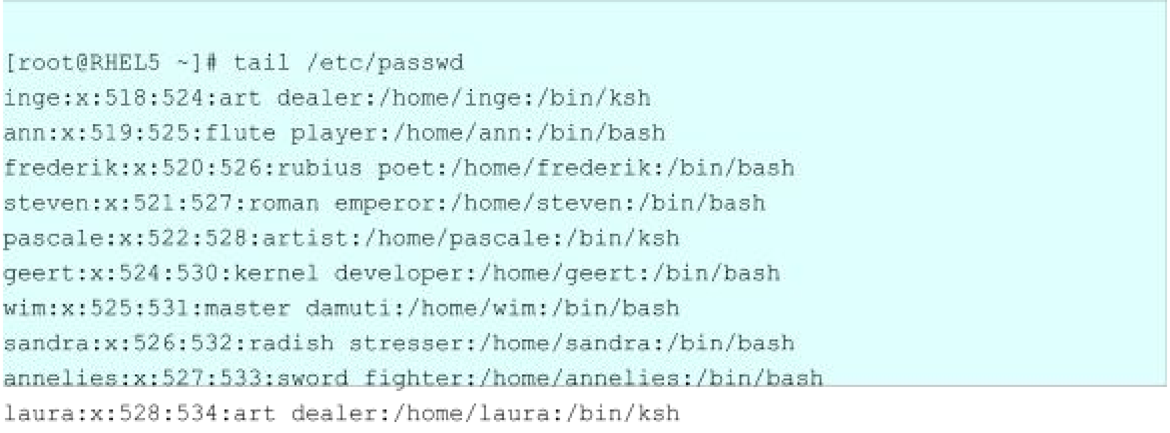
User management on any Unix can be done in three complimentary ways. You can use the **graphical** tools provided by your distribution. These tools have a look and feel that depends on the distribution. If you are a novice Linux user on your home system, then use the graphical tool that is provided by 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!

25.2.2. /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
```

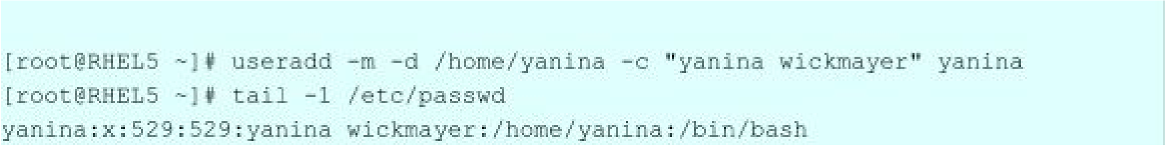
As you can see, this file contains seven columns separated by a colon. The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory, and the login shell.

25.2.3. 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).

25.2.4. 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).

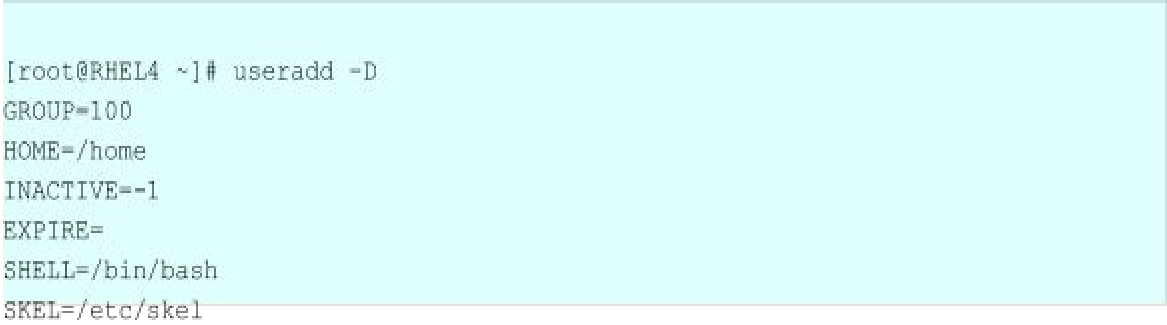


```
[root@RHEL5 ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina
[root@RHEL5 ~]# tail -1 /etc/passwd
yanina:x:529:529:yanina wickmayer:/home/yanina:/bin/bash
```

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

25.2.5. /etc/default/useradd

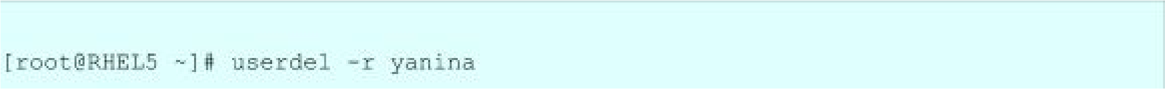
Both Red Hat Enterprise Linux and Debian/Ubuntu have a file called **/etc/default/useradd** that contains some default user options. Besides using cat to display this file, you can also use **useradd -D**.



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

25.2.6. userdel

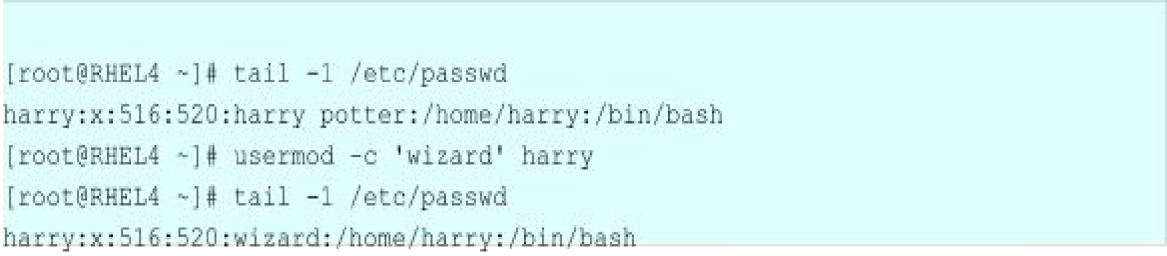
You can delete the user yanina with **userdel**. The -r option of userdel will also remove the home directory.



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

25.2.7. 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.

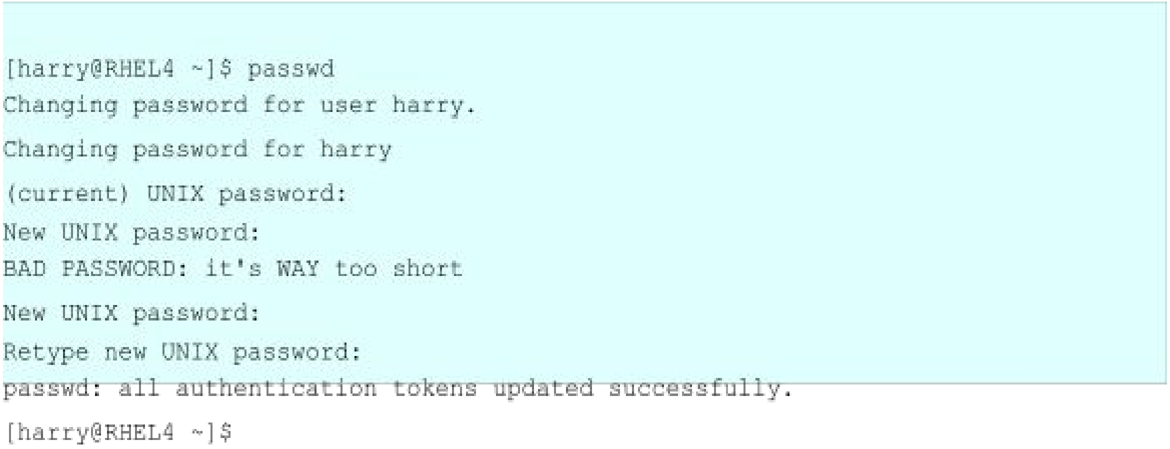


```
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:harry potter:/home/harry:/bin/bash
[root@RHEL4 ~]# usermod -c 'wizard' harry
[root@RHEL4 ~]# tail -1 /etc/passwd
harry:x:516:520:wizard:/home/harry:/bin/bash
```

25.3. passwords

25.3.1. 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.

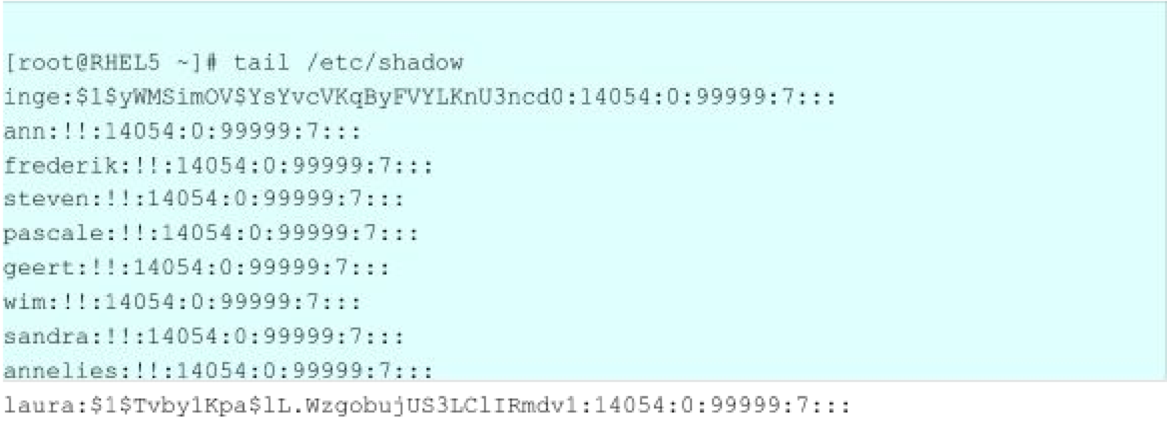


```
[harry@RHEL4 ~]$ passwd
Changing password for user harry.
Changing password for harry
(current) UNIX password:
New UNIX password:
BAD PASSWORD: it's WAY too short
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[harry@RHEL4 ~]$
```

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.

25.3.2. /etc/shadow

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.



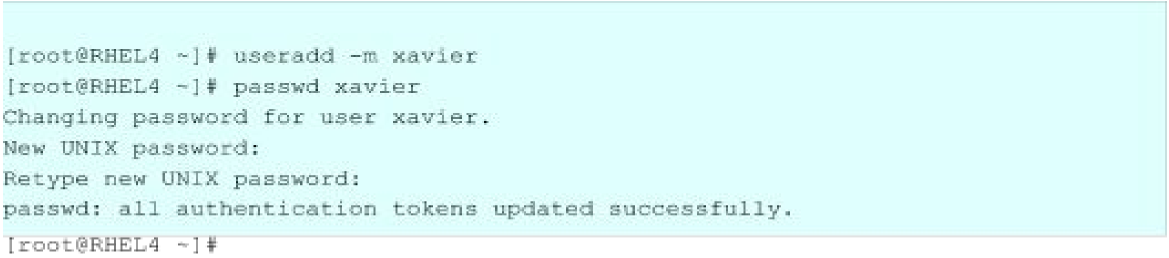
```
[root@RHEL5 ~]# tail /etc/shadow
inge:$1$yWMSimOV$YsYvcVKqByFVYLKnU3ncd0:14054:0:99999:7:::
ann:!!:14054:0:99999:7:::
frederik:!!:14054:0:99999:7:::
steven:!!:14054:0:99999:7:::
pascale:!!:14054:0:99999:7:::
geert:!!:14054:0:99999:7:::
wim:!!:14054:0:99999:7:::
sandra:!!:14054:0:99999:7:::
annelies:!!:14054:0:99999:7:::
laura:$1$Tvby1Kpa$lL.WzgobujUS3LClIRmdv1:14054:0:99999:7:::
```

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.

25.3.3. password encryption

encryption with passwd

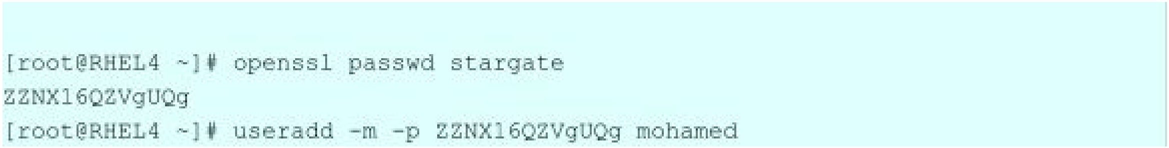
Passwords are stored in an encrypted format. This encryption is done by the **crypt** function. The easiest (and recommended) way to add a user with a password to the system is to add the user with the **useradd -m user** command, and then set the user's password with **passwd**.



```
[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 ~]#
```

encryption with openssl

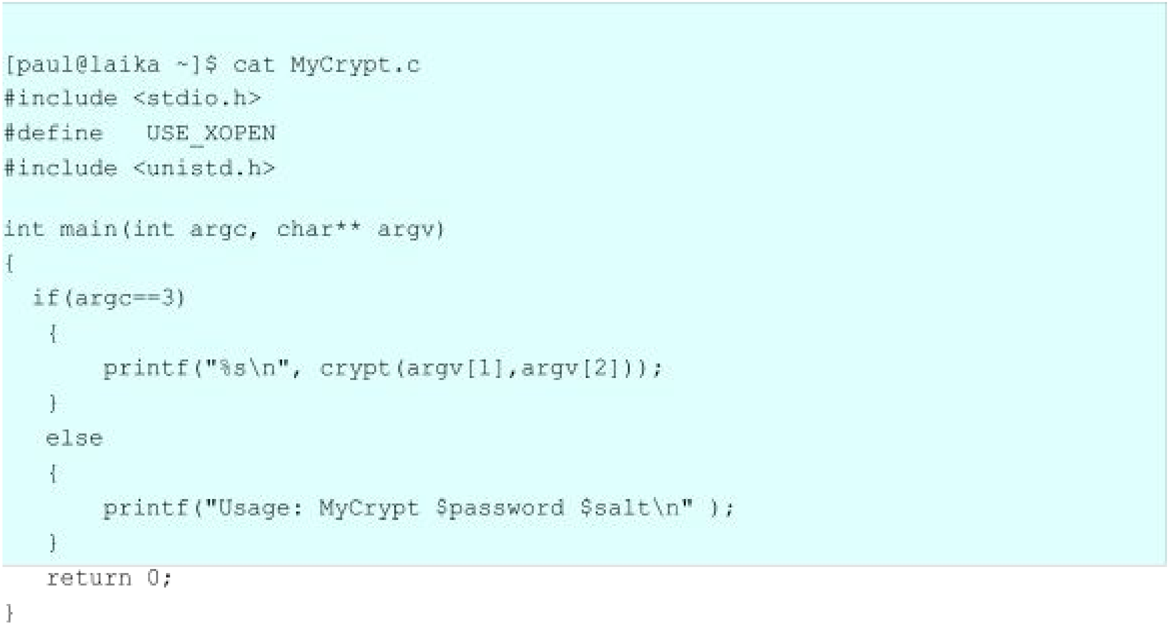
Another way to create users with a password is to use the -p option of useradd, but that option requires an encrypted password. You can generate this encrypted password with the **openssl passwd** command.



```
[root@RHEL4 ~]# openssl passwd stargate
ZZNX16QZVgUQg
[root@RHEL4 ~]# useradd -m -p ZZNX16QZVgUQg mohamed
```

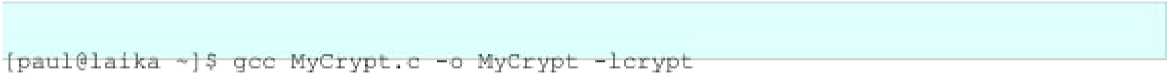
encryption with crypt

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



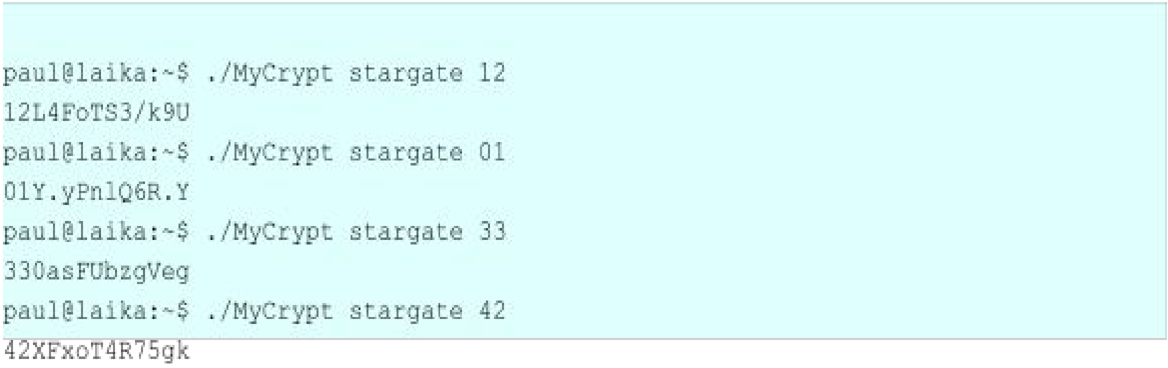
```
[paul@laika ~]$ cat MyCrypt.c
#include <stdio.h>
#define   USE_XOPEN
#include <unistd.h>

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@laika ~]$ gcc MyCrypt.c -o MyCrypt -lcrypt
```

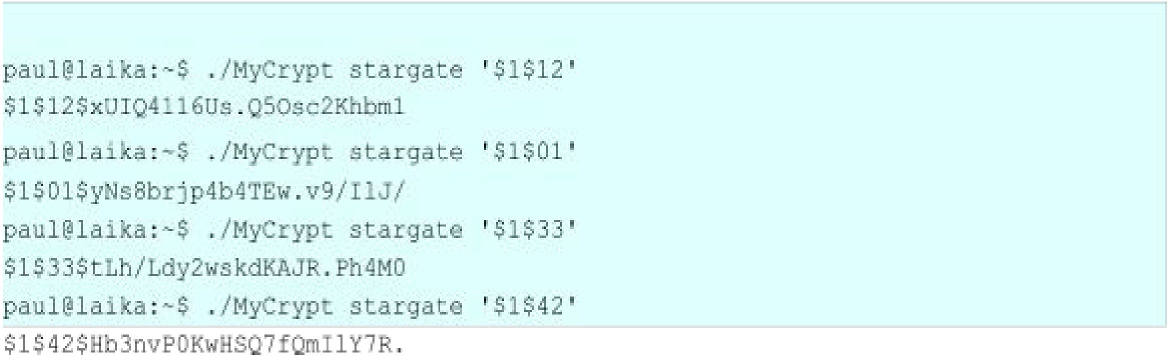
To use it, we need to give two parameters to MyCript. The first is the unencrypted password, the second is the salt. The salt is used to perturb the encryption algorithm in one of 4096 different ways. This variation prevents two users with the same password from having the same entry in **/etc/shadow**.



```
paul@laika:~$ ./MyCrypt stargate 12
12L4FoTS3/k9U
paul@laika:~$ ./MyCrypt stargate 01
01Y.yPnlQ6R.Y
paul@laika:~$ ./MyCrypt stargate 33
330asFUbzgVeg
paul@laika:~$ ./MyCrypt stargate 42
42XFxoT4R75gk
```

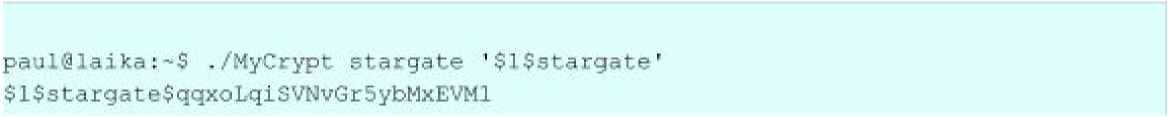
Did you notice that the first two characters of the password are the **salt**?

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@laika:~$ ./MyCrypt stargate '$1$12'
$1$12$xUIQ4116Us.Q5Osc2Khbm1
paul@laika:~$ ./MyCrypt stargate '$1$01'
$1$01$yNs8brjp4b4TEw.v9/IlJ/
paul@laika:~$ ./MyCrypt stargate '$1$33'
$1$33$tLh/Ldy2wskdKAJR.Ph4M0
paul@laika:~$ ./MyCrypt stargate '$1$42'
$1$42$Hb3nvP0KwHSQ7fQmIlY7R.
```

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!

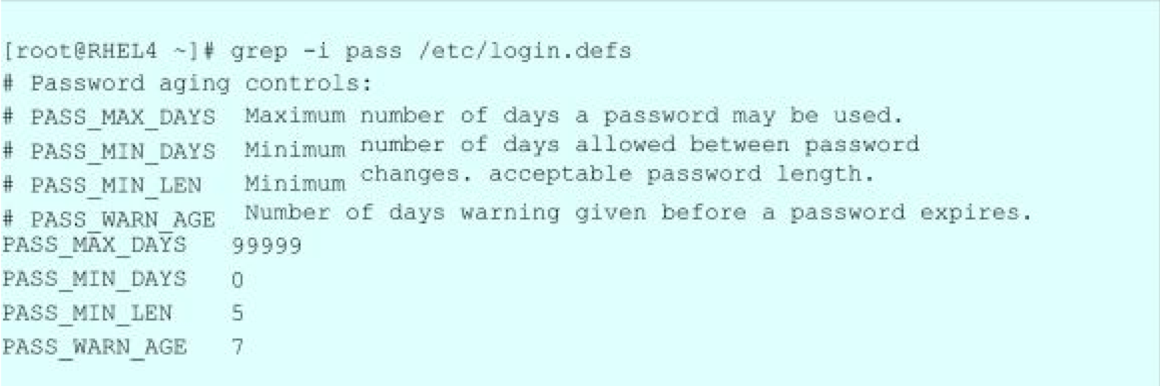


```
paul@laika:~$ ./MyCrypt stargate '$1$stargate'
$1$stargate$qqxoLqiSVNvGr5ybMxEVM1
```

25.3.4. password defaults

/etc/login.defs

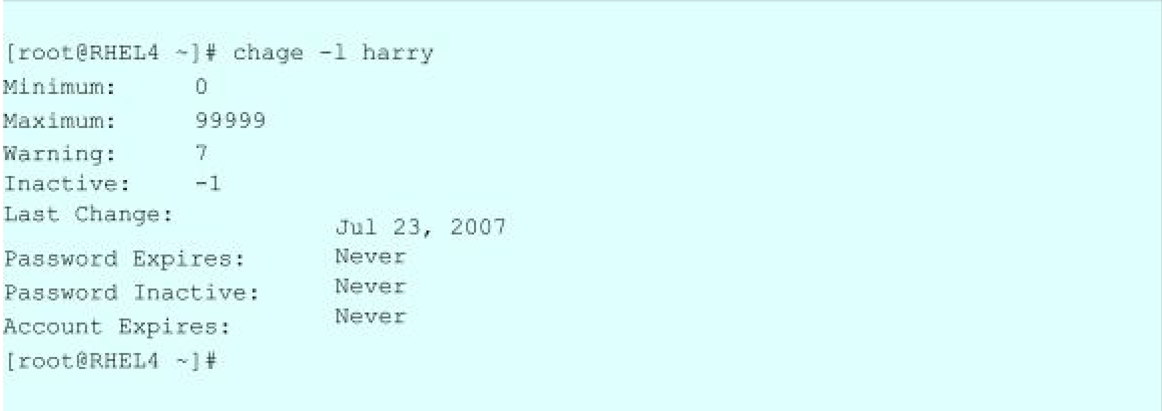
The **/etc/login.defs** file contains some default settings for user passwords like password aging and length settings. (You will also find the numerical limits of user ids and group ids and whether or not a home directory should be created by default).



```
[root@RHEL4 ~]# grep -i pass /etc/login.defs
# Password aging controls:
# PASS_MAX_DAYS  Maximum number of days a password may be used.
# PASS_MIN_DAYS  Minimum number of days allowed between password
# PASS_MIN_LEN   Minimum changes. acceptable password length.
# PASS_WARN_AGE  Number of days warning given before a password expires.
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   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@RHEL4 ~]# chage -l harry
Minimum:        0
Maximum:        99999
Warning:        7
Inactive:       -1
Last Change:
                        Jul 23, 2007
Password Expires:       Never
Password Inactive:      Never
Account Expires:        Never
[root@RHEL4 ~]#
```

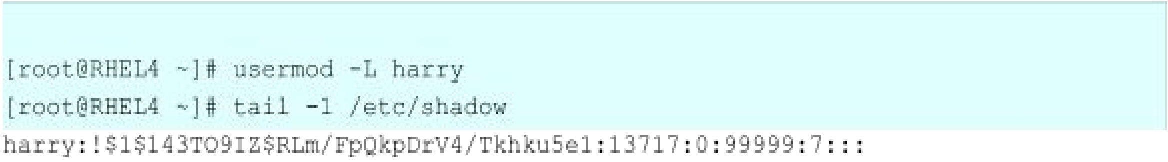
25.3.5. disabling a password

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.

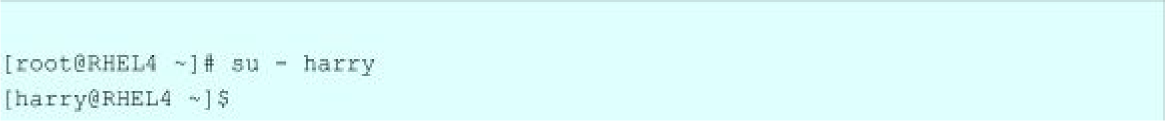
Besides vi (or vipw) you can also accomplish this with **usermod**.

The first line in the next screenshot will disable the password of user harry, making it impossible for harry to authenticate using this password.



```
[root@RHEL4 ~]# usermod -L harry
[root@RHEL4 ~]# tail -1 /etc/shadow
harry:!$1$143TO9IZ$RLm/FpQkpDrV4/Tkhku5e1:13717:0:99999:7:::
```

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



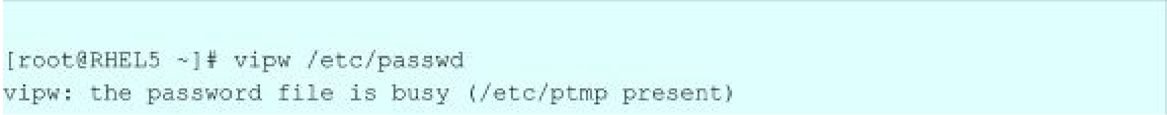
```
[root@RHEL4 ~]# su - harry
[harry@RHEL4 ~]$
```

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

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

25.3.6. 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.



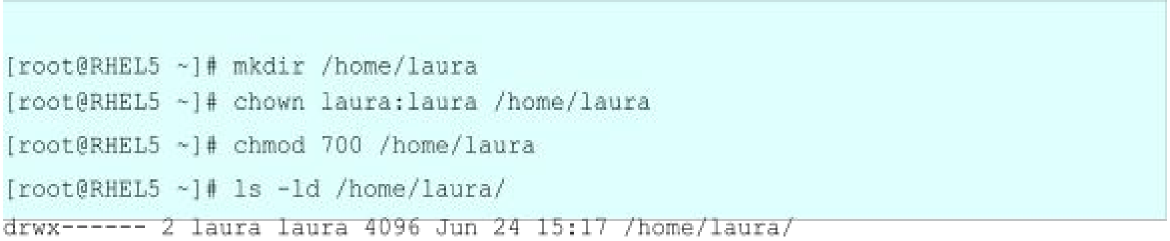
```
[root@RHEL5 ~]# vipw /etc/passwd
vipw: the password file is busy (/etc/ptmp present)
```

25.4. home directories

25.4.1. 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).

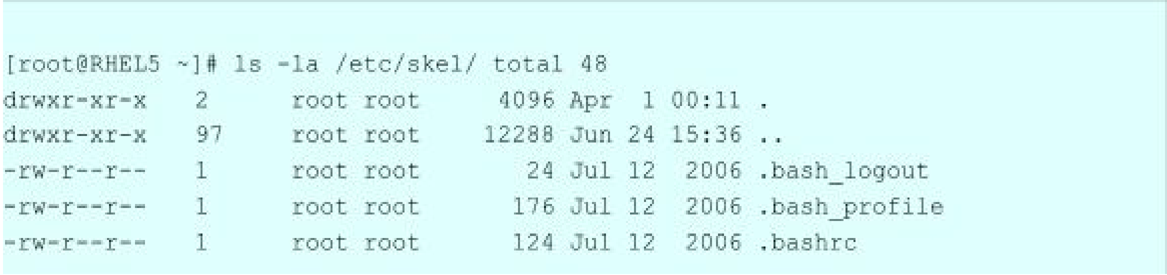
A less easy way is to create a home directory manually with **mkdir** which also requires setting the owner and the permissions on the directory with **chmod** and **chown** (both commands are discussed in detail in another chapter).



```
[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/
```

25.4.2. /etc/skel/

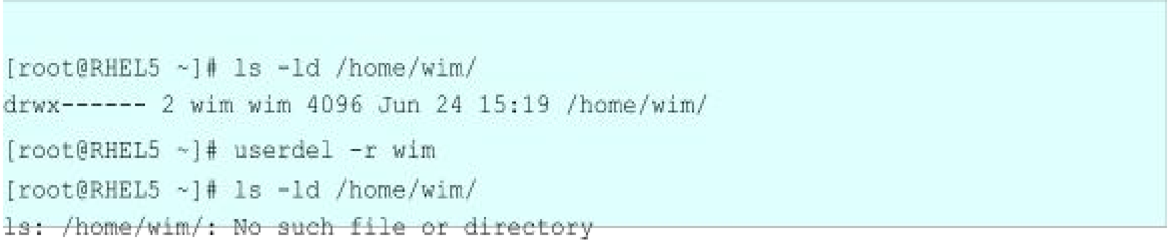
When using **useradd** the **-m** option, the **/etc/skel/** directory is copied to the newly created home directory. The **/etc/skel/** directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way **/etc/skel/** serves as a default home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

25.4.3. deleting home directories

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

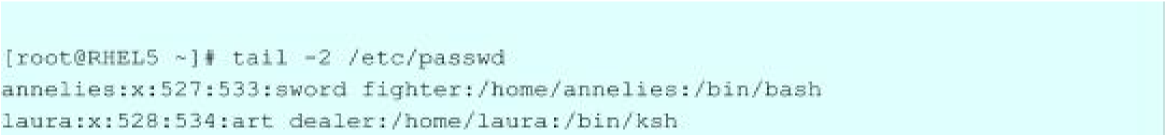


```
[root@RHEL5 ~]# ls -ld /home/wim/
drwx------ 2 wim wim 4096 Jun 24 15:19 /home/wim/
[root@RHEL5 ~]# userdel -r wim
[root@RHEL5 ~]# ls -ld /home/wim/
ls: /home/wim/: No such file or directory
```

25.5. user shell

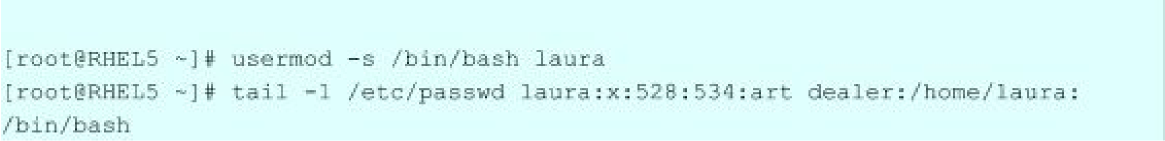
25.5.1. login shell

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
```

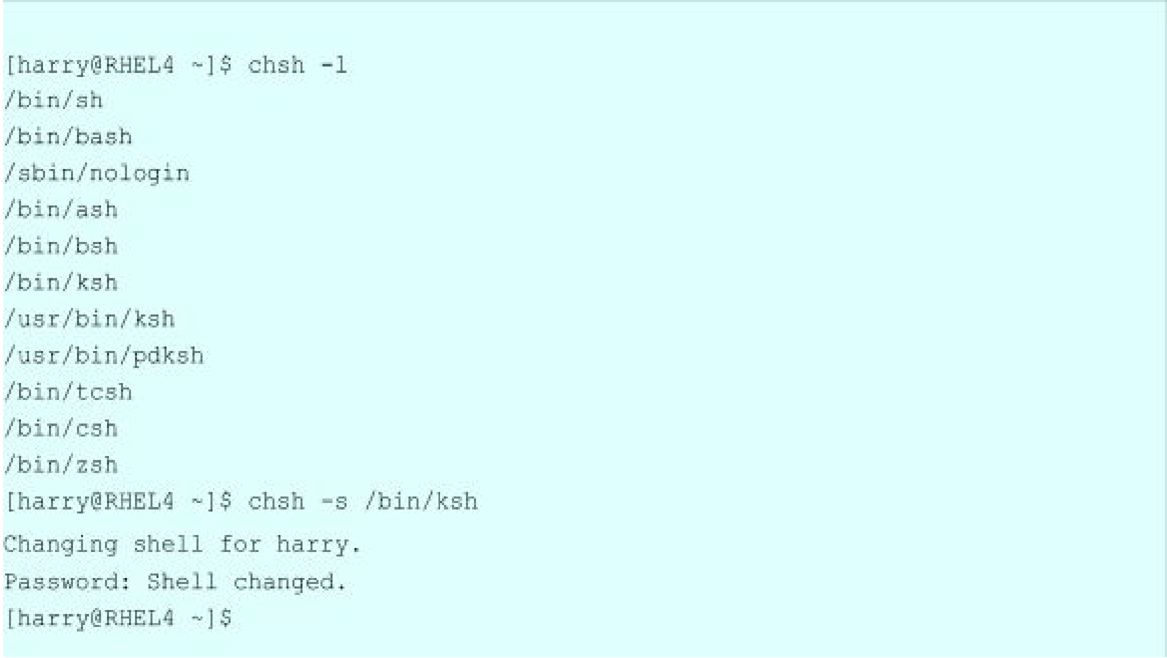
You can use the usermod command to change the shell for a user.



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

25.5.2. 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.

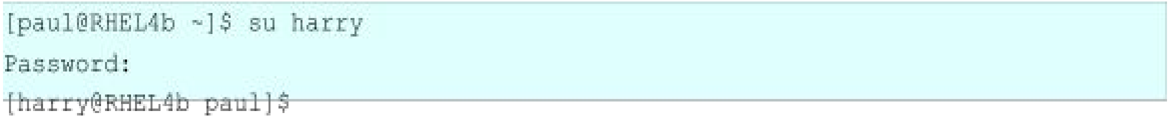


```
[harry@RHEL4 ~]$ chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/bin/ash
/bin/bsh
/bin/ksh
/usr/bin/ksh
/usr/bin/pdksh
/bin/tcsh
/bin/csh
/bin/zsh
[harry@RHEL4 ~]$ chsh -s /bin/ksh
Changing shell for harry.
Password: Shell changed.
[harry@RHEL4 ~]$
```

25.6. switch users with su

25.6.1. su to another user

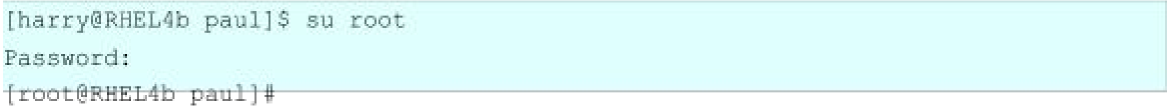
The **su** command allows a user to run a shell as another user.



```
[paul@RHEL4b ~]$ su harry
Password:
[harry@RHEL4b paul]$
```

25.6.2. su to root

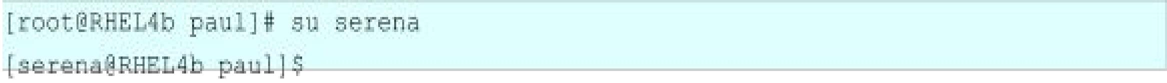
Yes you can alsu **su** to become **root**, when you know the **root password**.



```
[harry@RHEL4b paul]$ su root
Password:
[root@RHEL4b paul]#
```

25.6.3. su as root

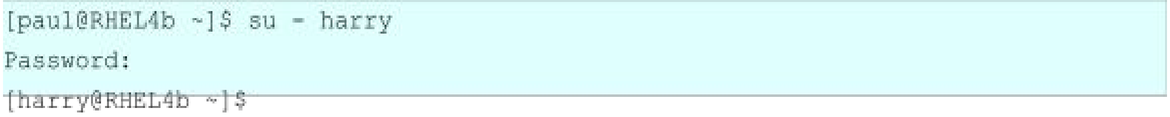
Unless you are logged in as **root**, running a shell as another user requires that you know the password of that user. The **root** user can become any user without knowing the user's password.



```
[root@RHEL4b paul]# su serena
[serena@RHEL4b paul]$
```

25.6.4. su - $username

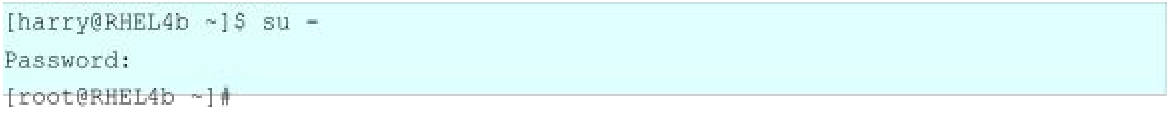
By default, the **su** command maintains the same shell environment. To become another user and also get the target user's environment, issue the **su -** command followed by the target username.



```
[paul@RHEL4b ~]$ su - harry
Password:
[harry@RHEL4b ~]$
```

25.6.5. su -

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



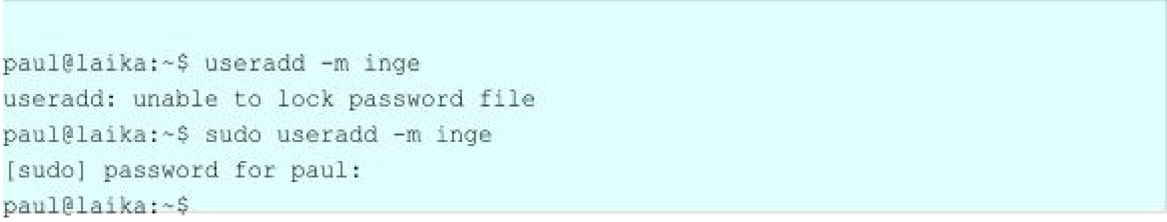
```
[harry@RHEL4b ~]$ su -
Password:
[root@RHEL4b ~]#
```

25.7. run a program as another user

25.7.1. about sudo

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).

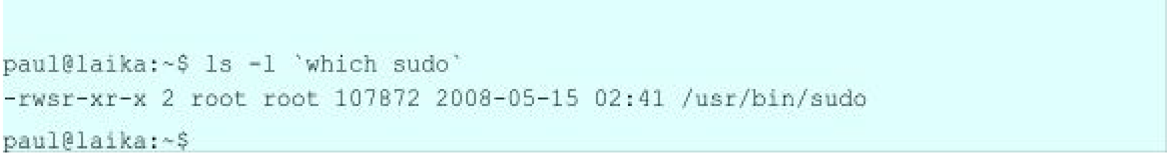
The screenshot below shows the usage of **sudo**. User **paul** received the right to run **useradd** with the credentials of **root**. This allows **paul** to create new users on the system without becoming **root** and without knowing the **root password**.



```
paul@laika:~$ useradd -m inge
useradd: unable to lock password file
paul@laika:~$ sudo useradd -m inge
[sudo] password for paul:
paul@laika:~$
```

25.7.2. setuid on sudo

The **sudo** binary has the **setuid** bit set, so any user can run it with the effective userid of root.



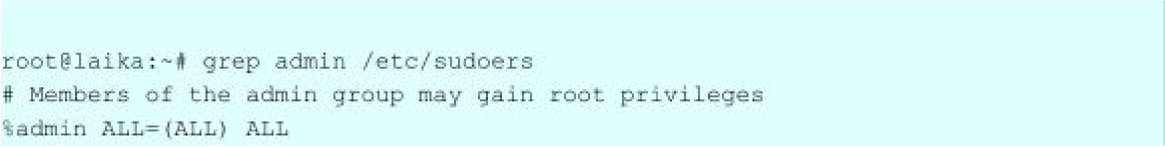
```
paul@laika:~$ ls -l `which sudo`
-rwsr-xr-x 2 root root 107872 2008-05-15 02:41 /usr/bin/sudo
paul@laika:~$
```

25.7.3. visudo

Check the man page of **visudo** before playing with the /etc/sudoers file.

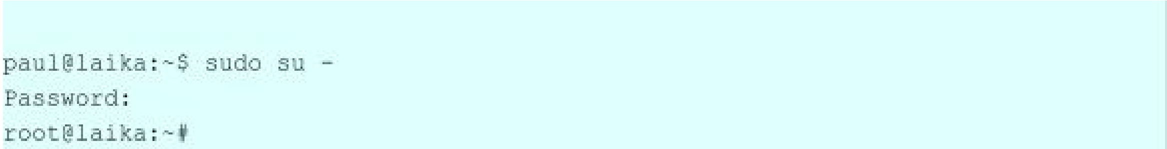
25.7.4. sudo su

On some Linux systems like Ubuntu and Kubuntu, 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:~#
```