Everything about Linux' User- and Filemanagement

Lukas Prokop

20. April 2009

Inhaltsverzeichnis

Basics – Users and Information

1 Who I am

On UNIX systems you are identified by your username and your password. On command line interfaces your name will generally appear as the first argument in the prompt (something like user@host path%). If you are logged in as root the percentage sign % will change to a rhomb #. However... this is system-specific stuff and I want to make references to the Linux distributions grml[1] and ubuntu[2]. Most of the commands are available on all Linux[3] systems, because they all try to be POSIX-compatible.

[1] Linux distribution based on Debian; Live CD for sysadmins, texttool-users and geeks
[2] well-regarded Debian based distribution; focused on usability
[3] GNU/Linux

1.1 whoami

Alright... I hope you were able to login. Sometimes you have to manage different accounts and log in with different usernames in different windows. You will lose the overview. In this case the UNIX command *whoami* will be helpful.

```
# CLI
root@localhost   # whoami
root
```

1.2 passwd

One of the most important things is changing your password. In difference to other OS' like Windows, Linux does not show any stars or other signs while typing your password. In the first moment it will confuse you, but you will realize that your partner (behind you) should not know the length of your password. That's only a security issue. Furthermore there is an algorithm which checks the security level of the password. So passwords like eg. "abc" are not accepted.

```
# CLI
root@localhost   # passwd
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
Password unchanged
Enter new UNIX password:
Retype new UNIX password:
You must choose a longer password
Enter new UNIX password:
Retype new UNIX password:
Bad: new password is too simple
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

2 My name is root and I'm your God

2.1 su

Alright... the most important user in a UNIX system is root. The name referes to a root, because user root is the user with UID zero and he is the only one, who can control the whole system. After installing your operating system, you will interact as root in

general, to configure all the application stuff.

I am root – I'm allowed to

But one person is also allowed to have several usernames. Maybe you want to set up two different accounts: "work" and "private". And you can also have to different accounts like "user" and "root". Basically root always exists. There is no way to delete him.

So of course you have to able to change accounts. It's very frustrating to reboot your computer, log in as root, install a program, reboot again and log in as the user back again. There is a much more simple way: *su* ("superuser"). This command lets you change your User ID. To get back to your previous account, use the keys Ctrl+D.

```
# CLI
user@localhost   % su
Password:
root@localhost /home/user #
user@localhost   %
```

After that action you can install programs (eg. with APT on Debian systems) or edit the user-configuration file /etc/passwd (see section 7). But there are a lot of other options, you can use with *su* . I will not describe all of the following examples, because some of them should explain themselve.

```
# CLI
user@localhost   % su -c 'echo $UID'
Password:
0
user@localhost   %
```

In this example su executes the command after -c with the login shell of the superuser. If we look into the /etc/passwd, it says /bin/zsh. So the command "echo $UID" will be executed by the zsh as root. And this returns the integer zero (the UID of root).

You can also append an other username to the command, to execute the command not as superuser (but the other username).

```
# CLI
user@localhost   % su
Password:
root@localhost /home/luki # echo $PATH
/sbin:/bin:/usr/bin:/usr/local/sbin:/usr/games
root@localhost /home/luki # pwd
/home/luki
root@localhost /home/luki #
user@localhost   % su -
Password:
root@localhost   # echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/games
root@localhost   # pwd
/root
root@localhost   #
user@localhost   % su -l
Password:
root@localhost   # echo $PATH
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/games
root@localhost   # pwd
/root
root@localhost   #
luki@localhost   %
```

What happened here? Well, if we use *su* we will get root permission, but actually this is not really a root session. We will keep user's working directory (pwd) and will keep user's PATH-environment. If we use *su -* , you will get the real root environment. But it is a standalone. So you can use exactly this command or (if you want to add additional options) you have to use *su -l* . In this environment the $PWD and $PATH will be different. User root is allowed to execute the programs in /usr/sbin (that's the difference between the two $PATH).

```
# CLI
user@localhost   % su -s /bin/bash otheruser Password:
otheruser@localhost:/home/user$
```

The option s of the *su* command is intended to define the shell you would like to use. If you don't specify one, *su* will find one from different source. The manpage tells in which the shell will be find.

- The shell specified with –shell
- If –preserve-environment is used, the shell specified by the $SHELL environment variable
- The shell indicated in the /etc/passwd entry for the target user

- /bin/sh if a shell could not be found by any above method

You can get all possible login shell with:

```
# CLI
user@localhost   % cat /etc/shells # /etc/shells: valid login shells
/bin/csh /bin/sh /usr/bin/es /usr/bin/ksh /bin/ksh /usr/bin/rc
/usr/bin/tcsh ... etc
```

You should realize that the superuser has to absolute control over the system. So sharing the password will lead to a compromised system. With Rootkits it will be possible to take over your system without your knowledge. So be aware to choose a difficult root password and don't write it down digital.

2.2 sudo

So we talked about the problem to become a superuser and work with this special accounts. But probably you don't want to give your friend the root password, but you want to allow him to install programs. In this case sudo will help you.

2.3 ubuntu su

Ubuntu has a special concept concering root permissions. Ubuntu is used by a lot of noobs and so they don't want anybody to surf the web with root privileges. They deactivated the superuser, which exists technically, but is not accessable (because of an – always wrong – password). So ubuntu users can execute everything as root by using *sudo* , but each time they have to add *sudo* explicitly, so that they are aware of the fact, that they are superuser. If an ubuntu user really needs a root shell (eg. because has he has to do a lot of administrative stuff for the next 20 minutes), he can access the root shell via *sudo -s* .

```
# CLI
user@localhost   % su
Password:
su: Authentication failure
1 user@localhost   % su -s /bin/bash Password:
root@localhost   #
```

2.4 root + BSD = toor

If you read toor backwards, you will get root. If you want to log in as root, the system has to get the root shell. But that shell can be damaged and so you won't be able to work as root. And because you are not root, you also cannot repair the damaged shell.

In this case BSD has created the toor-shell, which allows you to become a superuser without loading the root shell.

3 User data

Which data does the system store about you?

4 User configuration files

Where does the system store data about us?

4.1 /etc/passwd

4.2 /etc/shadow

4.3 /etc/passwd

4.4 /etc/profile

5 Filemanagement

6 Problems

6.1 Reseting a user

```
# CLI
#!/bin/bash

mkdir /root
cp /etc/skel/.* /root
chmod 755 /root
chown -R root:root /root
```

6.2 adduser

As root you can add new users with *adduser* .

```
# CLI
root@localhost /etc # adduser
Adding user ‘test’ ...
Adding new group ‘test’ (1001) ...
Adding new user ‘test’ (1001) with group ‘test’ ...
Creating home directory ‘/home/test’ ...
Copying files from ‘/etc/skel’ ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for test
Enter the new value, or press ENTER for the default
Full Name []: Admin Nerd
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [y/N] y
```

6.3 id

You can use *id* to get information about the current user.

```
# CLI
user@localhost /etc % id
uid=1000(user) gid=1000(user) groups=22(user), \
24(cdrom),25(floppy),29(audio),44(video),1000(user)
```

So what's that UID, GID and group-stuff?

If you want to give a user the permission to access a file, you can change that with *chmod* . But if you have 1000 users, it will get very difficult for you. So you can combine all users in one group and give this group the permissions. This is the main advantage of groups.

UIDs and GIDs are identifiers. Each file has several flags, which contain further information.

6.4 logname

This command returns the username you have used to login.

```
# CLI
user@localhost /etc % logname
user
user@localhost /etc % echo $LOGNAME
user
user@localhost /etc % su
Password:
root@localhost /etc # logname
root
```

So logname returns the value of system-variable LOGNAME.