Laboratory Work №5.
Account management.
Managing access rights to files and directories.

5.1 Linux, like any unix-like system, is not only multitasking, but also multi-user. This operating system allows multiple users to work with it simultaneously. But the system should somehow find out which user is working at the moment. For these purposes there are two concepts in Linux - ***accounts*** and ***authentication***, which are parts of one mechanism.

A user account is a user-specific information stored in special files. Information is used by Linux to authenticate the user and assign access rights to the user.

Authentication is a system procedure that allows Linux to determine which user is logged on. All information about the user is usually stored in the files */etc/passwd* and ***/etc/group***.

/etc/passwd – this file contains information about users. The entry for each user takes one line:

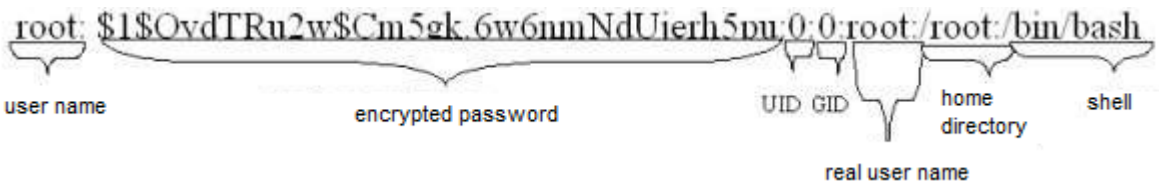

Figure 5.1

Username – name used by the user for all prompts of the login type when authenticating in the system.

Encrypted password – usually a user password or the symbol '!', Hashed under the irreversible MD5 algorithm, in cases when the user's interactive login to the system is prohibited.

UID – Numeric *user ID.* The system uses it to distribute permissions to files and processes.

GID –Numeric *group ID*. Group names are located in the */etc/group* file. The system uses it to distribute permissions to files and processes.

Real username – used for administrative purposes, as well as commands such as finger (getting information about the user through the network).

Home directory – full path to the user's home directory.

Shell – The command shell that the user uses during the session. For normal operation, it must be specified in the shell log file **/etc/shells**.

/etc/group – This file contains information about the groups to which the users belong:

Figure 5.2

Group name is the name used for the convenience of using programs such as ***newgrp***.

Encrypted password - used when changing groups with the command newgrp. There may be no password for groups.

GID is a numeric group id. The system uses it to distribute rights to files and processes.

Users included in several groups - In this field, those users whose default (in the ***/etc/passwd*** file) is assigned to another group are displayed separated by commas.

Today, storing passwords in the ***passwd*** and ***group files*** is considered unreliable.

The newer versions of Linux use the so-called shadow password files - ***shadow*** and ***gshadow***. The rights to them are assigned in such a way that even reading these files without superuser rights is impossible. It should be noted that the normal functioning of the system when using shadow files implies the presence of the ***passwd*** and ***group files*** at the same time. When using shadow passwords in ***/etc/passwd*** and /***etc/group***, the '***x***' character is set instead of the password itself, which is an indication that the password is stored in ***/etc/shadow*** or ***/etc/gshadow***.

The shadow file stores protected user information, and also provides mechanisms for the aging of passwords and accounts. Here is the structure of the shadow file:

Figure 5.3

a - user name;

b - encrypted password - hashing algorithms are used, usually MD5 or the symbol '!', In cases when the user's interactive login to the system is prohibited;

c - the number of days since the last password change, starting from January 1, 1970;

d - number of days before the password can be changed;

e - number of days after which the password must be changed;

f - number of days, for how many users will start to warn that the password is out of date;

g - number of days after the password expires for account lockout;

h - days, counting from January 1, 1970, when the account will be blocked;

i - reserved field;

The **gshadow file** also imposes additional functionality, coupled with the secure storage of group passwords. It has the following structure:

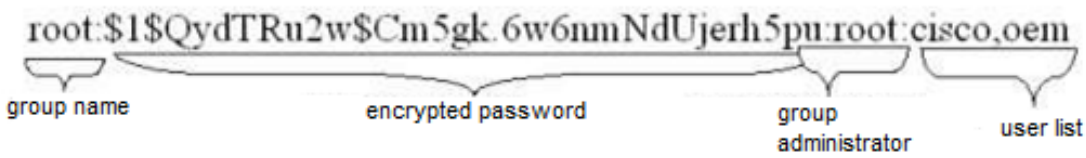

Figure 5.4

Group name is the name used for the convenience of using programs such as ***newgrp***.

Encrypted password - used when changing a group using the ***newgrp*** command. There may be no password for groups.

Group administrator is a user who has the right to change the password with ***gpasswd***.

User list - This field displays comma separated users who have a different group assigned to them by default (in the ***/etc/passwd*** file).

5.2 In Linux, in addition to ordinary users, there is one (and only one) user with unlimited rights. The identifiers of this user's **UID** and **GID** are always 0. Its name is usually root, but it can be easily changed (or several symbol names with the same **GID** and **UID** are created), since the value for applying unrestricted access rights has only **GID** 0. For the root user, the access rights to files and processes are not checked by the system. When using the root account, you must be extremely careful. There is always the possibility to destroy the system.

5.3 Linux uses a developed system of distribution of rights to users. But for the exact identification of a user, one name is not enough in terms of security. That is why a password is used, an arbitrary set of characters of arbitrary length, usually limited only by the encryption methods used.

Today, in most versions of Linux passwords are encrypted using the 3DES and MD5 algorithms (obsolete, now SHA512). When the 3DES algorithm is reversible, that is, such a password can be decrypted, MD5 is an irreversible transformation. Passwords encrypted using the 3DES algorithm do not apply when using shadow files to store passwords.

When authenticating, the password entered by the user is encrypted in the same way as the original one, and then the encrypted copies are compared. If they are the same, then authentication is considered successful.

Given the daily increasing security requirements, Linux has the ability to use hidden passwords.

The ***/etc/passwd*** and ***/etc/group*** files are readable to all users, which is a pretty big security flaw in the system. That is why in modern versions of Linux, it is preferable to use hidden passwords.

Such passwords are located in the files ***/etc/shadow*** and ***/etc/gshadow***, for user and group passwords respectively.

5.4 The ***login*** command starts the interactive session in the system. It checks the correctness of entering the user name and password, changes the directory to home, builds up the environment and runs the shell. The login command is usually not run from the command line - it's usually done by console managers-for example, ***getty*** or ***mgetty***.

The command ***su*** (switch user) allows you to change the user **ID** already during the session. Its syntax is simple: ***su username***, where username is the user name that will be used.

After that, the program will ask for the password. If the password is correctly entered, ***su*** will run a new shell with user rights specified by ***su*** and assign the session **IDs** to the session. If the user name is omitted, the ***su*** command uses the name root.

```
[student@ns student]$ su root
Password:
[root@ns student]#_
```

When using the ***su*** command as root, it does not normally ask for a password. The ***newgrp*** command is similar in its capabilities to ***su***, with the difference that a group change occurs.

The user must be included in the group, which is specified in the ***newgrp*** command line. When using ***newgrp*** as root, it never asks for a password. The command syntax is similar to the syntax of the ***su*** command: ***newgrp*** groupname, where **groupname** is the name of the group to which the user changes the current one.

The ***passwd*** command is a tool for changing passwords in Linux. To change your password, just type ***passwd*** at the command line:

```
[student@ns student]$ passwd
Changing password for student
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully
[student@ns student]$_
```

To change the group password and group management, use the gpasswd command. To change the password, type ***gpasswd*** **GROUPNAME** at the command prompt. You will be able to change the password only if you are the administrator of the group. If the password is not empty, then for the group members, the ***newgrp*** call does not require a password, and the group members must enter the password. The group administrator can add and remove users using the **–a** and **–d** options, respectively.

Use the **-r** option to delete the group password. If the password is not set, only the group members can use the ***newgrp*** command to enter the group. By specifying the **-R** option, you can deny access to the group by using the ***newgrp*** command (however, this does not apply to group members). The system administrator (root) can use the **-A** option to assign to the administrator group.

```
gserg@ADM:/$ chage -l gserg
Last password change                    : May 03, 2007
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
```

The superuser can also use other parameters, such as:

–d date (in system date format, for example DD.MM.YYYY) - sets the date of the last password change by the user.

–E date - set the expiration date of the user account.

–I N - set the number of days of inactivity N from the time the password res before the

account is locked.

–m N - specifies the minimum number of days (N) between password changes.

–M N - specifies the maximum number of days (N) between password changes.

–W N - specifies the number of days for which the warning about password expiration will be issued.

1. For each object in the Linux file system, there is a set of permissions that determines the interaction of the user with this object. Such objects can be files, directories, and also special files (for example, devices) - that is, in fact, any object of the file system. Since every object in Linux has an owner, the access rights apply to the owner of the file. They consist of a set of 3 groups of three attributes:

• reading (r), writing (w), executing (x) for the owner;

• Read, write, execute for the owner group;

• Read, write, execute for all others.

Such rights can be represented by a short entry:

rwxrwxrwx - read, write and execute permissions for all

rwxr-xr-x - the entry is allowed only for the owner of the file, and read and execute for all.

rw-rw-r-- - the entry is allowed for the owner of the file and the group of the file owner, and reading is for everyone.

This distribution of rights allows you to flexibly manage the resources available to users.

2. Access rights also apply to directories. They mean:

r - if the right to read from the directory is set, you can see its contents with the ***ls*** command.

w - if set to write to the directory, the user can create and delete files from the current directory. And delete a file from the directory user can even if it does not have write permission to the file. There is an opportunity to correct this situation. I'll talk about this later.

x - if the execution right is set to the directory, then the user has the right to go to such directory with commands like ***cd***.

Thus, it becomes possible to create so-called "hidden" directories when it is not possible to get a list of files, but a user who knows the exact file name can copy it from the "hidden" directory.

3. To distribute access rights in Linux, there are many commands. The main ones are ***chmod, chown*** and ***chgrp*****.**

The ***chmod*** command (Change MODe) changes the permissions of the file. To use this command, you must also have the rights of the file owner or the root authority. The syntax of the command is:

Chmod mode filename,

where

filename - the name of the file whose permissions are changed;

mode - the permissions to install on the file. Access rights can be written in 2 versions - symbolic and absolute.

In the symbolic form, the use of the ***chmod*** command will look like this:

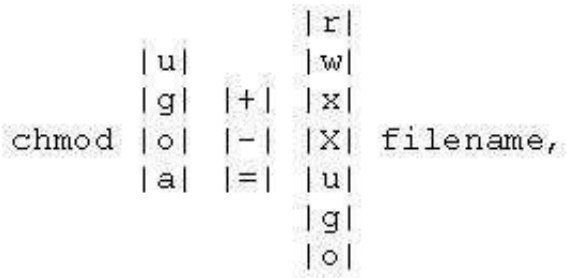

Figure 5.5

where:

u,g,o,a – setting permissions for the user, group, other users, and all access rights groups, respectively.

+,-,= – add, delete, set the resolution accordingly.

r,w,x,X,u,g,o – the right to read, write, execute, execute if there is such a right for any of the access groups, the same as for the owner, same as for the group, same as for the other users.

filename - the name of the file whose rights are changed.

View permissions installed on the file using the ***ls*** command with the ***-l*** option:

```
[student@ns  student]$  ls -l lesson5.txt
-rw------- 1 student student 39 Nov 19 15:17 lesson5.txt
[student@ns student]$ chmod g+rw lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rw-rw---- 1 student student 39 Nov 19 15:18 lesson5.txt
[student@ns student]$ chmod o=u lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rw-rw-rw- 1 student student 39 Nov 19 15:18 lesson5.txt
[student@ns student]$ chmod o-w lesson5.txt
[student@ns student]$ ls -l lesson5.txt
-rw-rw-r-- 1 student student 39 Nov 19 15:19 lesson5.txt
[student@ns student]$_
```

To use absolute mode, you need to provide access rights to the file in the form of 3 binary groups. For example:

rwx r-x r-- will be: **111 101 100**

Now convert each binary group to an 8-digit number:

111 – 7, 101 – 5, 100 – 4 .

To set the file for such rights, run the following command:

[student@ns student]$ ls -l lesson5.txt

-rw-rw-r-- 1 student student 39 Nov 19 15:19 lesson5.txt

[student@ns student]$chmod 754 lesson5.txt

[student@ns student]$ ls -l lesson5.txt

-rwxr-xr-- 1 student student 39 Nov 19 15:19 lesson5.txt

[student@ns student]$_

Task for trainees: try changing the permissions to file lesson5.txt and setting the following: ***r w x***

r- - r - - (744), r - - - w- - - x(421), - - x - w - r - -(124).

Also invite them to do the same in symbolic form.

Command ***chown*** (*CHange* OWNer) – allows you to change the owner of the file. To use this command, you must either have the rights of the current file owner or the root authority. The command syntax is simple:

chown username:groupname filename , where

username – the username of the new file owner;

groupname – group name - new file owner;

filename – the name of the file whose owner changes.

The name of the group in the syntax of the command can be omitted, then only the owner of the file will be changed.

Command ***chgrp*** used to change the owner-group of a file. The syntax is:

chgrp groupname filename,

where:

groupname – *the name of the group to which the file belongs*

filename – the name of the file to be modified

Keep in mind that ***chown*** and ***chmod*** can only be used by the user who owns the file and root, and the ***chgrp*** command is the owner of the file, the group owner of the file and root.

4. There are a few special rights that can be installed on files and directories. We will consider one now. This is the so-called sticky bit (attachment bit). In the first versions of Unix, this bit was used to force the system to leave an image of its code in memory when the program is running. Then the next time you access the program, it was consumed much less time to start it, because reading the code from the device was no longer required.

For files, and today in Linux, the former value of this bit remains. But for directories this attribute has acquired a new meaning. If the sticky bit is installed on the directory, then only the user who owns the file can delete the files from that directory, and only if he has the right to write to the file. The owner group and other users, even if they have write permissions to the file, will not be able to remove it when installed to the sticky bit directory.

chmod +t filename

Task for laboratory work

In the course of the work, it is necessary to study theoretical information related to user administration, as well as to carry out practical tasks and answer the quiz described below.

1. Read the contents of the files:
   - /etc/passwd,
   - /etc/shadow,
   - /etc/group.
2. Create the following groups:
   - workers,
   - teachers,
   - students.

3. Create user user_[variant]_ N, where N =1, 2, .., 5, uid account must be equal ot 1000+N.

Users with N equal to 1 and 2 add to the group of workers manually making changes to the configuration file.

After adding users, check the /etc /group file for errors. Users with N equal to 3, 4 and 5 are added to the group students using the administration commands *.

If you have any questions about using a particular key, use the man command to get help:

man [command name].

Check the results.

4. Create user teacher_[variant].

In the comment to the account there must be your first and last name.

The uid of the account must be 3000. Add the user to the teachers group.

5. For all users, set the password using the ***passwd*** command.

6. Create a labs directory in the root directory. In it, create the library and tests directories.

7. Create files book_[student name]_N and put it in library.

8. Create text file **test_[student name]**, and put it into **tests** folder. Files must contain a script: to create a user *user[variant number]* and to set password *pass[variant]*. Give the right to execute the file to the **Students** group users.

9. In the directory **labs** create file **list**, that must contain a list of files in directory ***/*etc**.

10. Give the right to ***modify*** the file only to the user teacher_[variant number], but workers - ***readonly***.

11. Configure access rights to the library and tests directory, so that users of the **teachers** group can modify and create files there, and students of **students** have read access.

Quiz:

1. Why are the passwords not stored explicitly in the configuration files?

2. Why is not it recommended to perform daily operations using a root account?

3. What is the difference between the mechanisms for obtaining special ***su*** and ***sudo*** privileges?

4. When you run the ***ls -la*** command, you get the result:
-rw-r-x-r-- 1 den factory 4464 30 May 2008 text.txt. What does it mean?

5. How do I set permissions on a directory and all objects in it?