Kyiv Professional College of Communications

Computer Engineering Cycle Commission

**REPORT**

**LABORATORY WORK №10**

in the discipline: "Operating Systems"

**Topic: "Special directories and files in Linux"**

Completed by:

Students of the RPZ-93b group

**Team 1**: Usenko B.A,

Melnichuk M.A

The teacher:

Sushanova V.S

Kyiv 2022

**The purpose of the work:**

1. Gaining practical skills in working with the Bash command shell.

2. Familiarity with special directories and files in Linux.

**Material support:**

1. Computer type IBM PC.

2. Windows 7.

3. Virtual machine - Virtual Box (Oracle).

4. GNU / Linux operating system - CentOS.

5. Cisco Network Academy website netacad.com and its online Linux courses.

***Tasks for preliminary preparation***

***Prepared material by student Melnichuk M.A.***

1. Read brief theoretical information for laboratory work and make a small dictionary of basic English terms on the purpose of team assignments and their parameters.

|  |  |
| --- | --- |
| **Term** | **Purpose** |
| setuid permission | Permission when the binary file is run as the owner of the file, not as the user who executed it. |
| chmod u+s file | To add the setuid permission symbolically |
| chmod u-s file | To remove the setuid permission symbolically |
| setgid permission | The setgid permission is similar to setuid, but it makes use of the group owner permissions. |
| chmod g+s <file|directory> | To add the setgid permission symbolically |
| chmod g-s <file|directory> | To remove the setgid permission symbolically |
| sticky bit | Sticky bit permission is used to prevent other users from deleting files that they do not own in a shared directory. |
| links | посилання |
| inode number*.* | A unique identification number of file. |
| symbolic link | A soft link, is simply a file that points to another file. |

2. Based on the considered material give answers to the following questions:

2.1. When the setuid permission is set on an executable binary file (a program) the binary file is run as the owner of the file, not as the user who executed it. This permission is set on a handful of system utilities so that they can be run by normal users, but executed with the permissions of root, providing access to system files that the normal user doesn't normally have access to.

The *setgid* permission on a file is very similar to *setuid*; it allows a user to run an executable binary file in a manner that provides them additional (temporary) group access. The system allows the user running the command to effectively belong to the group that owns the file, but only in the setgid program.

2.2. The *sticky bit* permission is used to prevent other users from deleting files that they do not own in a shared directory. Recall that any user with write permission on a directory can create files in that directory, as well as delete any file in the directory, even if they do not own the file!

The sticky bit permission allows for files to be shared with other users, by changing write permission on the directory so that users can still add and delete files in the directory, but files can only be deleted by the owner of the file or the root user.

A good example of the use of sticky bit directories would be the /tmp and /var/tmp directories. These directories are designed as locations where any user can create a temporary file.

Because these directories are intended to be writable by all users, they are configured to use the sticky bit. Without this special permission, users would be able to delete any files in this directory, including those that belong to other users.

3. Study Cisco Academy Online Course Materials:

- NDG Linux Essentials (Chapter 18 all Topics)

4. Take the NDG Linux Essentials course on the following topics:

- Chapter 18 Exam

***Progress Prepared material by student Usenko B.A.***

* 1. Initial work in CLI mode in Linux OS of the Linux family:
  2. Start the Ubuntu\_PC virtual machine ***(if you perform LR tasks through the netacad academy)***
  3. Study all the examples of commands presented in the laboratory work of the course ***NDG Linux Essentials:***
* ***Lab 18: Special Directories and Files***
  1. Create a table of commands studied in paragraph 2 in the following form:

|  |  |
| --- | --- |
| **Command name** | **Its purpose and functionality** |
| ls -ld /tmp  ls -ld /var/tmp | Using the -d option for the ls command lists directory information; combined with the -l option it shows ownership and permissions for the directory files. List the details of the /tmp and /var/tmp directories. |
| ln | create a hard link |
| rm | remove files |
| ln -s source softlink | The -s option for the ln command creates a symbolic link instead of a hard link. Create a symbolic link to the source file and view the details of both files: |

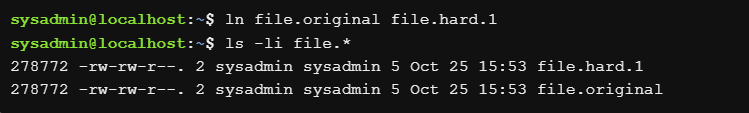
**Answers to control questions:**

***Prepared material by students Melnichuk M.A., Usenko B.A.***

1. To create hard links, the ln command is used with two arguments. The first argument is an existing file name to link to, called a target, and the second argument is the new file name to link to the target.



When the ln command is used to create a hard link, the link count number increases by one for each additional filename:



The peculiarity of hard links is that they directly point to the index descriptor number and, accordingly, such names can only point to files within the same file system, ie on the same media on which is the directory containing this name.

2.Another type of link that can be created is known as a symbolic link or soft link. Symbolic links do not increase the link count of files with which they are linked.

Symbolic link files have their own inode and type of file. Instead of linking and sharing an inode, they link to the file name. Unlike hard links, soft links can be linked to directories and can cross devices and partitions to their targets.

The -s option for the ln command creates a symbolic link instead of a hard link. Create a symbolic link to the source file and view the details of both files:

ln -s source softlink

ls -li source softlink

3. Hard links & Soft links:

|  |  |
| --- | --- |
| **HARD LINKS** | **SOFT LINKS** |
| **Hard links don’t have a single point of failure. I**f you have five files hard linked together, then deleting any four of these files would not result in deleting the actual file contents. | Symbolic links, have a single point of failure: the original file. If the original file is removed, then any files linked to it fail. |
| Sometimes it can be difficult to know where the hard links to a file exist. If you see a regular file with a link count that is greater than one, you can use the find command with the -inum search criteria to locate the other files that have the same inode number. | **Soft links are easier to see.** Soft links are much more visual, not requiring any extra commands beyond the ls command to determine the link. |
| Since each file system (partition) has a separate set of inodes, hard links cannot be created that attempt to cross file systems | **Soft links can link to any file.** |
| Hard links cannot be created on directories. The reason for this limitation is that the operating system itself uses hard links to define the hierarchy of the directory structure. | **Soft links can link to a directory.** |

*4*.If you have five files hard linked together, then deleting any four of these files would not result in deleting the actual file contents. Sometimes it can be difficult to know where the hard links to a file exist. If you see a regular file with a link count that is greater than one, you can use the find command with the -inum search criteria to locate the other files that have the same inode number.

If the original file is removed, then any files linked to it fail. Soft links are easier to see. Soft links are much more visual, not requiring any extra commands beyond the ls command to determine the link.

**Conclusion:**

We gained practical skills in working with the Bash shell, got acquainted with special directories and files in Linux.