

OPERATING SYSTEMS LAB



LAB MANUAL # 05

Instructor: Muhammad Abdullah

Orakzai Semester Fall 2021

Course Code: CL2006

**Fast National University of Computer and Emerging
Sciences, Peshawar**

Department of Computer Science

Contents

	1. NANO
Editor.....	1 2. Links
.....	1 1.

Hard Links.....	2 2.
Soft Links.....	2 3.
More Less.....	4
1. more command.....	4 2.
less Command.....	5 4.
Searching.....	5
1. find command.....	5 2.
locate Command.....	6 5. File
Permissions.....	7
Changing file permissions with 'chmod' command	
.....	9 1. Absolute (Numeric) Mode
.....	9 2. Symbolic Mode
.....	10 1. Permission to
Others	11 2. Permission to Group
.....	12 3. Permission to
User.....	12 6. Directory
Permissions.....	13 1.
Permission to User.....	13 2.
Permission to Group	14 3.
Permission to Others	14 7. File
or Directory Ownership.....	15 8.
GNU Compiler Collection	16
How To Write C Program In Ubuntu.....	16
How To Write C++ Program In Ubuntu.....	17 9.
How to execute any command in Linux (ubuntu) through C Program.....	19
.....	10. References
.....	19

CL220 - OPERATING SYSTEM LAB

1. NANO Editor

There are many text editors available for Linux. At the moment you will have access to the nano editor. Nano is an advanced text editor provided by GNU. Simply typing *nano* on the shell will give you the editor. You may also start your nano by explicitly mentioning the file you want to work on. This file may be already existing or you may be creating a new one.

Near the end of your screen you will see a list of shortcuts. The ones which you should get yourself familiar with are as such:

CTRL+X for exit

CTRL+O for saving

CTRL+W for searching

CTRL+K for cutting

CTRL+U for pasting

CTRL+C for displaying cursor position

Other commands are listed at the bottom of the text-editor window.

2. Links

Links are the equivalent of Shortcuts in Windows. The syntax of creating a link to a file or directory.

A link in UNIX/linux is a pointer to a file. Like pointers in any programming languages, links in UNIX are pointers pointing to a file or a directory. Creating links is a kind of shortcuts to access a file. Links allow more than one file name to refer to the same file, elsewhere.

There are two types of links:

1. Soft Link or Symbolic links

2. Hard Links

These links behave differently when the source of the link (what is being linked to) is moved or removed. Symbolic links are not updated (they merely contain a string which is the path name of its target); hard links always refer to the source, even if moved or removed.

For example, if we have a file **a.txt**. If we create a hard link to the file and then delete the file, we can still access the file using hard link. But if we create a soft link of the file and then delete the file, we can't access the file through soft link and soft link becomes dangling. Basically, hard link increases reference count of a location while soft links work as a shortcut (like in Windows).

1. Hard Links

- Each hard-linked file is assigned the **same Inode value** as the original, therefore they reference the same physical file location. Hard links more flexible and remain linked

even if the original or linked files are moved throughout the file system, although hard links are unable to cross different file systems.

- `ls -l` command shows all the links with the link column shows number of links. •

Links have actual file contents

- Removing any link, just reduces the link count, but doesn't affect other links. • Even if we change the filename of the original file then also the hard links properly work.

- We cannot create a hard link for a directory to avoid recursive loops. • If original file is removed then the link will still show the content of the file. • The size of any of the hard link file is same as the original file and if we change the

content in any of the hard links then size of all hard link files are updated. • The disadvantage of hard links is that it cannot be created for files on different file systems and it cannot be created for special files or directories.

- Command to create a hard link is:

```
$ ln [original filename] [link name]
```

2. Soft Links

- A soft link is similar to the file shortcut feature which is used in Windows Operating systems. Each soft linked file contains a **separate Inode value** that points to the original file. As similar to hard links, any changes to the data in either file is reflected in the other. Soft links can be linked across different file systems, although if the original file is deleted or moved, the soft linked file will not work correctly (called hanging link).
- `ls -l` command shows all links with first column value `l`? and the link points to original file.
- Soft Link contains the path for original file and not the contents.
- Removing soft link doesn't affect anything but removing original file, the link becomes "dangling" link which points to nonexistent file.
- A soft link can link to a directory.

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar

Page **3** of **21**

- Size of a soft link is equal to the name of the file for which the soft link is created. E.g If name of file is file1 then size of it's soft link will be 5 bytes which is equal to size of name of original file.
- If we change the name of the original file then all the soft links for that file become dangling i.e. they are worthless now.
- Link across file systems: If you want to link files across the file systems, you can only use symlinks/soft links.
- Command to create a Soft link is:

```
$ ln -s [original filename] [link name]
```

To practice, try out the following commands:

ln existingfile linkname

```
# ls
```

```
# touch blankfile
```

```
# mkdir blankdir
```

```
# ls
```

```
# ln blankdir dirpointer
```

```
# ln -s blankdir dirpointer
```

```
# ln blankfile filepointer
```

```
# cat < blankfile
```

```
# cat < filepointer
```

```
# echo "Hello dear" > filepointer
```

```
# cat < blankfile
```

```
# cat < filepointer
```

```
# ls
```

```
# ls -lh
```

Look carefully at the contents of both filepointer and dirpointer. Changing one will automatically change the other. Also look at how the directory pointers appear using ls -lh.

3. More | Less

Sometimes a user may give a command which generates so much output that it scrolls very fast. As a result, the top information simply flows out of the screen whereas only the

low-end information is visible. For example, use the following command *ls /etc -lh*

We can view the lost information using the more or less commands. Try both of them

first. *ls /etc -lh | more*

To quit, press q.

ls /etc -lh | less

The syntax of both above commands are such that we are specifying two commands in one go. The first command starts with ls, the second command starts with more or less. Both these commands are joined by the pipe symbol |.

With more, you are able to browse down the display 1-page at a time using spacebar. With less, you are able to browse up and down the display 1-line at a time using up and down arrow keys. An alternative is using output redirection that you have covered in last lab. Usage would be as such:

ls /etc -lh > temp.txt
nano temp.txt

1. more command

The more command is quite similar to the cat command, as it is used to display the file content in the same way that the cat command does. The only difference between both commands is that, in case of larger files, the more command displays screenful output at a time.

In more command, the following keys are used to scroll the page:

ENTER key: To scroll down page by line.

Space bar: To move to the next page.

b key: To move to the previous page.

Syntax:

more <file name>

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar

Page 5 of 21

e.g. more temp.txt

man printf |more

man scanf |more

2. less Command

The less command is similar to the more command. It also includes some extra features

such as 'adjustment in width and height of the terminal.' Comparatively, the more command cuts the output in the width of the terminal.

Syntax:

less <file name>

e.g. less temp.txt

man printf |less

man scanf |less

4. Searching

By default, searching for files or directories is performed using the find command.

1. find command

Find command is used to locate a specific by name or extension.

The syntax of find command is as such:

find <where> -name <what>

The following symbols are used after the find command:

(.) : For current directory name

(/) : For root

So, if I want to find all pdf files in / directory, I will give:

find / -name ".pdf"*

sudo find / -name ".pdf"*

To find file name use the following command (In Root Directory)

sudo find / -name prog

sudo find / -name prog.c

finding file in current directory

```
sudo find . -name prog.c
```

```
sudo find . -name prog
```

finding directory by name

```
sudo find . -name blankdir
```

Searching with file Extention

```
sudo find / -name "*.pdf"
```

```
sudo find . -name "*.c"
```

```
sudo find . -name "*.cpp"
```

```
sudo find . -name "*.txt"
```

find Desktop -name "*.txt" will find all txt files in Desktop directory.

2. locate Command

The locate command is used to search a file by file name. It is quite similar to find command; the difference is that it is a background process. It searches the file in the database, whereas the find command searches in the file system. It is faster than the find command. To find the file with the locates command, keep your database updated.

Syntax: locate <file name>

e.g. locate "*.pdf" , locate test4 (test4 is directory name)

(Explore **locate** command by yourself)

5. File Permissions

All files and directories in Linux have an associated set of owner permissions that are

used by the operating system to determine access. These permissions are grouped into **three sets of three bits**. The **sets** represent {owner, group, everyone else} whereas the **bits** represent {read, write, execute}. So overall, we have 9 permissions, as shown in the following table:

	Read	Write	Execute
Owner	1	1	1
Group	1	1	0
Others	0	0	1
	2^2	2^1	2^0

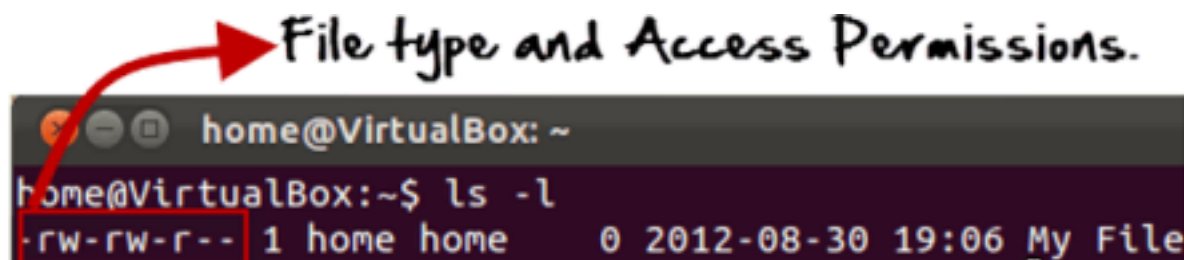
If we look at the first row for Owner, we see three 1's. Which specify that the Owner has read, write, and execute permissions on a file or directory. Since all of the bits are in allow mode, we can add them up together to get $2^2 + 2^1 + 2^0 = 4 + 2 + 1 = 7$. Similarly, the second row for Group, i.e., users within the same group as that of the file owner, have read and write, but no execute permissions. This will be translated only as $2^2 + 2^1 = 4 + 2 = 6$. And lastly, every other user can only execute files and have no permission to read or write to them. This will be translated as $2^0 = 1$. So, the file permissions would be 761. To give the permission of 761 to a file, we would use the command:

```
chmod 761 <filename>
```

```
chmod 777 <filename>
```

You can view the file permissions using the command:

```
ls -lh
```



```

muhammad@muhammad-VirtualBox:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 muhammad muhammad 33 09:50 14  مارچ file2.txt
-rw-rw-r-- 1 muhammad muhammad 0 16:13 22  مارچ hello.sh
drwxrwxr-x 2 muhammad muhammad 4096 19:22 4  اپریل test
muhammad@muhammad-VirtualBox:~/Desktop$

```

The characters are pretty easy to remember.

r = read permission
w = write permission
x = execute permission
- = no permission

The first part of the code is '**rw-**'. This suggests that the owner 'Home' can:

rw-
 ↓
 no execute permission

- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.



-rw-rw-r-- (-) show normal file.

drw-rw-r (d) show directory.

crw-rw-r (c) show character special file.

brw-rw-r (b) show binary special file.

Changing file permissions with 'chmod' command

We can use the '**chmod**' command which stands for '**change mode**'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world.

Syntax: *chmod permissions filename*

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

1. Absolute (Numeric) Mode

In this mode, file permissions are not represented as characters but a three-digit octal number.

The table below gives numbers for all for permissions types.

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--X
2	Write	-W
3	Execute + Write	-WX
4	Read	r--
5	Read + Execute	r-X
6	Read +Write	rW
7	Read + Write +Execute	rWX

Let's see the chmod permissions command in action.

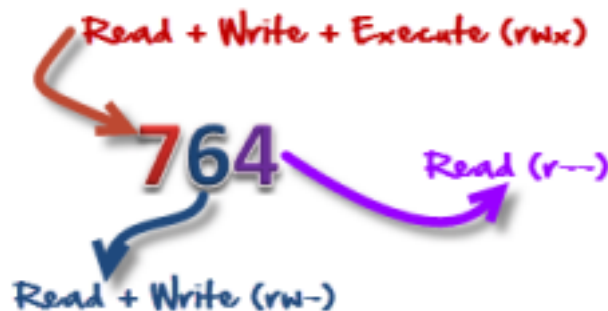
Checking Current File Permissions

```
ubuntu@ubuntu:~$ ls -l sample
-rw-rw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

chmod 764 and checking permissions again

```
ubuntu@ubuntu:~$ chmod 764 sample
ubuntu@ubuntu:~$ ls -l sample
-rwxrw-r-- 1 ubuntu ubuntu 15 Sep  6 08:00 sample
```

In the above-given terminal window, we have changed the permissions of the file 'sample' to '764'.



'764' absolute code says the following:

- Owner can read, write and execute
- Usergroup can read and write
- World can only read

This is shown as '-rwxrw-r--'

This is how you can change user permissions in Linux on file by assigning an absolute number.

2. Symbolic Mode

In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the Unix file permissions.

Operator

Description

+ Adds a permission to a file or directory - Removes the permission

= Sets the permission and overrides the permissions set earlier.

The various owners are represented as –

User Denotations	
u	user/owner
g	Group
o	Other
a	All

We will not be using permissions in numbers like 755 but characters like rwx.

1. Permission to Others

1) To allow others to execute file

`chmod o+x filename`

where **o** stands for other, **x** means executable permissions and **+** is used for adding permissions.

e.g. `chmod o+x testfile.txt`

`ls -l`

2) To give permission to others to write file

`chmod o+w filename`

e.g. `chmod o+w testfile.txt`

`ls -l`

where **w** stands for write permissions.

`chmod o+wx testfile.txt` -----> Give both permissions to others

ls -l

chmod o+rw testfile.txt -----> Give all permissions to users

ls -l

2) Take permission from others

chmod o-w testfile.txt -----> Take write permission from others

ls -l

chmod o-wx testfile.txt -----> Take write and executable permission from others

ls -l

chmod o-rwx testfile.txt -----> Take all permissions from users

2. Permission to Group

chmod o+w filename

e.g. chmod g+w testfile.txt

ls -l

chmod g-wx testfile.txt

ls -l

chmod o+w testfile.txt

3. Permission to User

chmod u+r testfile.txt

ls -l

chmod u+w testfile.txt

ls -l

chmod u+x testfile.txt

ls -l

chmod u+rw testfile.txt

ls -l

```
chmod u-rwx testfile.txt
```

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar
Page **13** of **21**

```
ls -l
```

To give permission to everybody or take permission from every body

```
chmod ug=rwx testfile.txt ----->To give all permissions to user and group
```

```
ls -l
```

```
chmod ugo=rwx testfile.txt ----->To give all permissions to user and group and others
```

```
ls -l
```

OR

```
chmod ugo+rwx testfile.txt----->To give all permissions to user and group and others
```

```
chmod ugo-rwx testfile.txt -----> To take all permissions to user and group and others
```

```
chmod a+rwx testfile.txt -----> a stand for all (Give all permissions to all) chmod a-rwx  
testfile.txt -----> (take all permissions from all)
```

```
chmod u+rw,g+rw,o+r testfile.txt
```

```
ls -l
```

This how you can change the permissions of a file using symbolic notations. , w, x are called symbolic notations.

6. Directory Permissions

1. Permission to User

```
ls -l
```

```
chmod u-w test ----->Take write permission from user
```

```
ls -l
```

```
chmod u+wx test ----->Give all permission to user
```

```
ls -l
```

```
chmod u=wx test ----->Give all permission to user
```

```
ls -l
```

chmod u-wrx test ----->Take all permissions from user

ls -l

cd test

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar

Page 14 of 21

bash: cd: test: Permission denied

chmod u+x test executable permission given to user

now

cd test

mkdir test4

mkdir: cannot create directory 'test4': Permission

denied cd ..

chmod u+w test

cd test

mkdir test4

ls -l

ls: cannot open directory '.': Permission

denied cd ..

chmod u+r test

chmod u+r test

cd test

2. Permission to Group

chmod g+wx test ----->Give all permission to

group ls -l

chmod g-wrx test ----->Take all permissions from group

ls -l

3. Permission to Others

chmod o+wx test ----->Give all permission to

other ls -l

chmod o-wrx test ----->Take all permissions from other

ls -l

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar
Page **15** of **21**

7. File or Directory Ownership

A file's owner can be changed using the command:

chown <username> <filename>

Where, <username> would be the username of any user on your system, and <filename> is the target to which this setting is going to apply to. **chown** stands for **change owner**.

ch-----> change

own----->owner

Examples:

sudo chown usman test4 Change the directory current owner to **usman**

ls-l

sudo chown usman file.txt Change the file current owner to **usman**

ls-l

In case you want to change the user as well as group for a file or directory use the command.

`chown user:group filename`

sudo chown usman:usman file.txt

ls-l

cat > test.txt -----> use this command you will get the following message.

“bash: test.txt: Permission denied”. This is because that currently working user is muhammad in my case and owner of the file is usman.

sudo chown usman:usman test4

ls-l

In case you want to change group-owner only, use the command

```
chgrp group_name filename
```

'**chgrp**' stands for change group.

```
sudo chgrp muhammad file.txt
```

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar

Page **16** of **21**

8. GNU Compiler Collection

Programs written in C on Linux are compiled using the gcc compiler. Programs written in C++ are compiled using the g++ compiler. Both these compilers are provided by GNU under the label GCC.

Any C program written for linux should have the extension of .c (e.g., hello.c), whereas a C++ program should have an extension of .cpp (e.g., hello.cpp). When compiling, the general syntax of command is as follows:

```
gcc first.c [-o first]
```

```
g++ first.cpp [-o first]
```

A breakdown is as such:

- gcc|g++ is the compiler itself
- first.c|first.cpp will be the filename with extension of your source code. • [] Anything inside this square brackets is an optional arguments. Do not write the square brackets themselves in your command.
- -o is the output filename. It is the argument for making an executable file with the name you specify. Without this, the source code will be compiled into an executable file named a.out first is the name of executable file which you pass to -o argument.

So, considering that we have a source code with the name first.c, and compile it with the above command, we can execute it using:

```
./first
```

Where ./ specifies the current directory, and first is the name of executable file which you passed as an argument to the compiler.

How To Write C Program In Ubuntu

- Open a text editor (gedit, VI). Command: `gedit prog.c`
- Write a C program. Example: `#include<stdio.h> int main(){ printf("Hello"); return 0;}` • Save C program with .c extension. Example: `prog.c`
- *Compile C program*. Command: `gcc prog.c -o prog`
- Run/ Execute. Command: `./prog`

1. Enter the command given below in a terminal window to open a [text editor](#).

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar
Page **17** of **21**

```
gedit prog.c
```

Here `prog.c` is the name of the program. After entering the above code, it will ask for the password.

2. Now write your program, for your convenience, I have given a sample program below, you can copy and paste it into the gedit text editor.

```
#include<stdio.h>

int main()

{

printf("\nThis is C Programming in Ubuntu\n\n");

return 0;

}
```

3. Now save and close the editor.
4. ***Compile the C program in Ubuntu as below.*** GCC compiler will compile our code.

```
gcc prog.c -o prog
```

5. If there will be no error in the program, then nothing will be shown. And if an error occurs, it will be shown. In case you get an error you have to open the text editor again by repeating steps 1 and remove it and again save and close the editor.

6. Enter the command given below to run the C program.

```
./prog
```

How To Write C++ Program In Ubuntu

- Open any text editor (gedit, vi). Command: `gedit progcpp.cpp`
- Write a C++ program. Example: `#include int main(){ cout<<"Hello world"; return 0;}` •
Save C++ program with .cpp extension. Example: `progcpp.cpp`
- Compile C++ program. Command: `g++ progcpp.cpp -o progcpp` •
Run/ Execute. Command: `./progcpp`

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar
Page **18** of **21**

We will follow same steps as C program we did.

1. Enter the command given below in the terminal window to open a text editor.

```
gedit progcpp.cpp
```

Now you can write your program, I have given an example

below. `#include<iostream>`

```
using namespace std;
```

```
int main()
```

```
{
```

```
cout<<"\nThis is C++ Programming in Ubuntu\n\n";
```

```
return 0;
```

```
}
```

2. Now save and close the editor.

3. To compile the C++ program in Linux. Enter the command given below in the terminal window.

```
g++ progcpp.cpp -o progcpp
```

4. Enter the command given below to run the C++ program.

```
./progcpp
```

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar
Page **19** of **21**

9. How to execute any command in Linux (ubuntu) through C Program

system() function is used to execute command of linux through c program. This system function is defined in **stdlib.h** header file. So, you are required to include this header file in your program.

The system() function is a part of the C/C++ standard library. It is used to pass the commands that can be executed in the command processor or the terminal of the operating system, and finally returns the command after it has been completed.

<stdlib.h> or <cstdlib> should be included to call this function.

Steps:

1. Write a program namely command and save it with .c extension. i.e. command.c
2. Write this command in terminal **gedit command.c** geditor will be opened, write your C program in this editor.

```
#include<stdio.h>
#include<stdlib.h>
void main()
```

```
{  
printf("\nThis is C Programming in Ubuntu\n\n");  
printf("\nHere in this program we will run linux commands\n\n");  
system("ls -l"); // this command will print all the contents of current directory in long format. }
```

3. Compile your c program using gcc compiler i.e. **gcc command.c -o command**.
4. After successful compilation execute you program from terminal and check the result.

10. References

<https://www.guru99.com/file-permissions.html>

Good Luck :)

Prepared By: Muhammad Abdullah Orakzai (Instructor CS) FAST-NUCES Peshawar