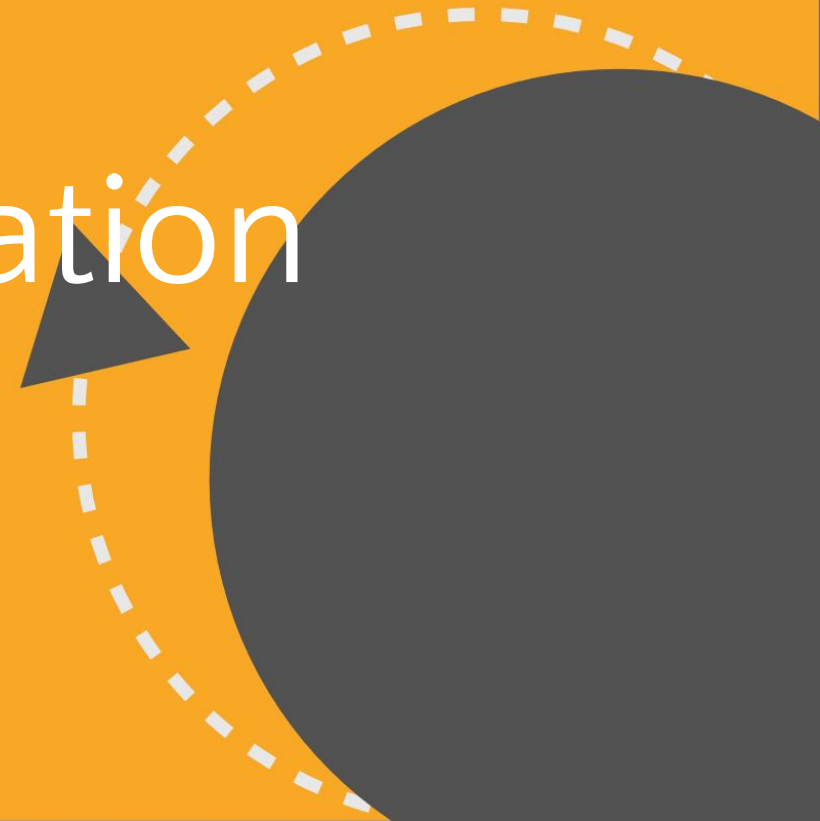




Open Source Community

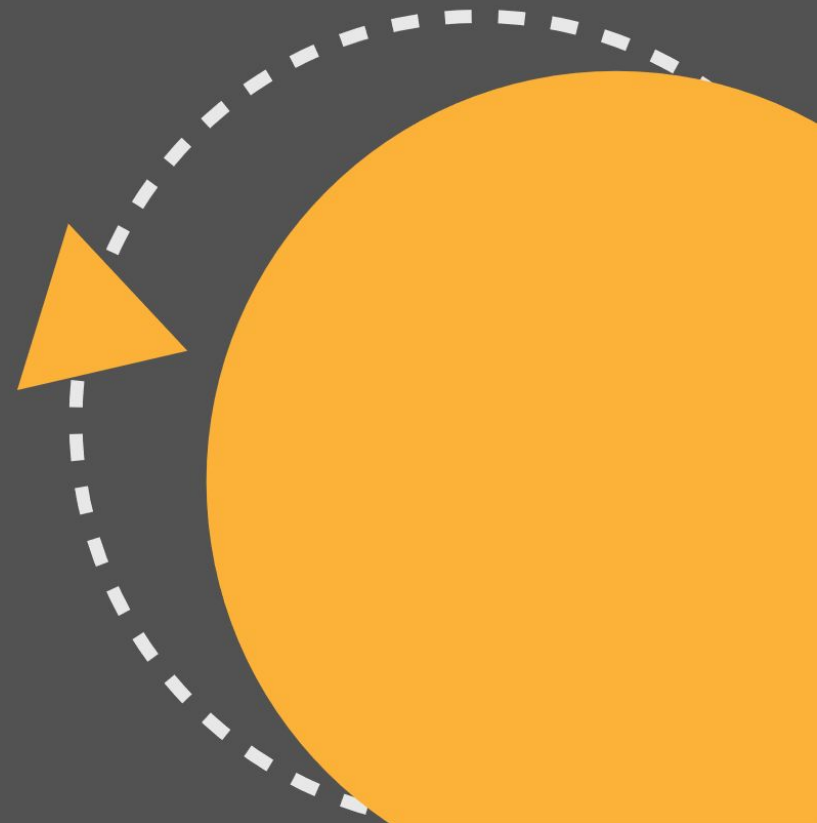
Files & Text Manipulation





Agenda;

-
- CLI Text Editors
 - Compile C++ with g++
 - Reading Files
 - I/O Streams and Redirection
 - File Permissions



GNU Nano

GNU Nano or Nano for short, is a text editor for Unix-like computing systems or operating environments using a command line interface(CLI).

To start using nano we can simply run it from the terminal by running the command `nano`.

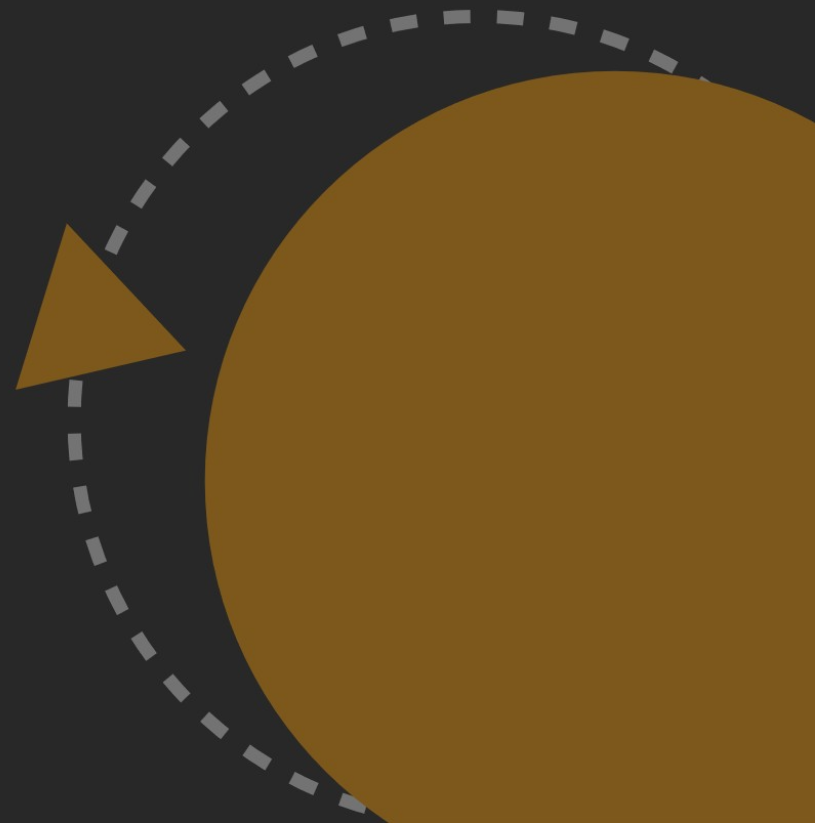
We can write whatever text we want and once we're done we can exit using the shortcut `Ctrl + x` (^X) , we'll be prompted to enter the name of the file to save as.

Example:

`nano file.txt`

A screenshot of the GNU nano 2.9.3 text editor running in a terminal. The interface has a dark background with light-colored text. At the top, there is a menu bar with options: File, Edit, View, Search, Terminal, and Help. Below the menu bar, the title bar shows "GNU nano 2.9.3" on the left and "New Buffer" on the right. The main editing area is currently empty. At the bottom of the screen, there is a status bar displaying various keyboard shortcuts and their functions in a grid-like format: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify; ^X Exit, ^R Read File, ^\ Replace, ^U Uncut Text, ^T To Spell.

Reading Files



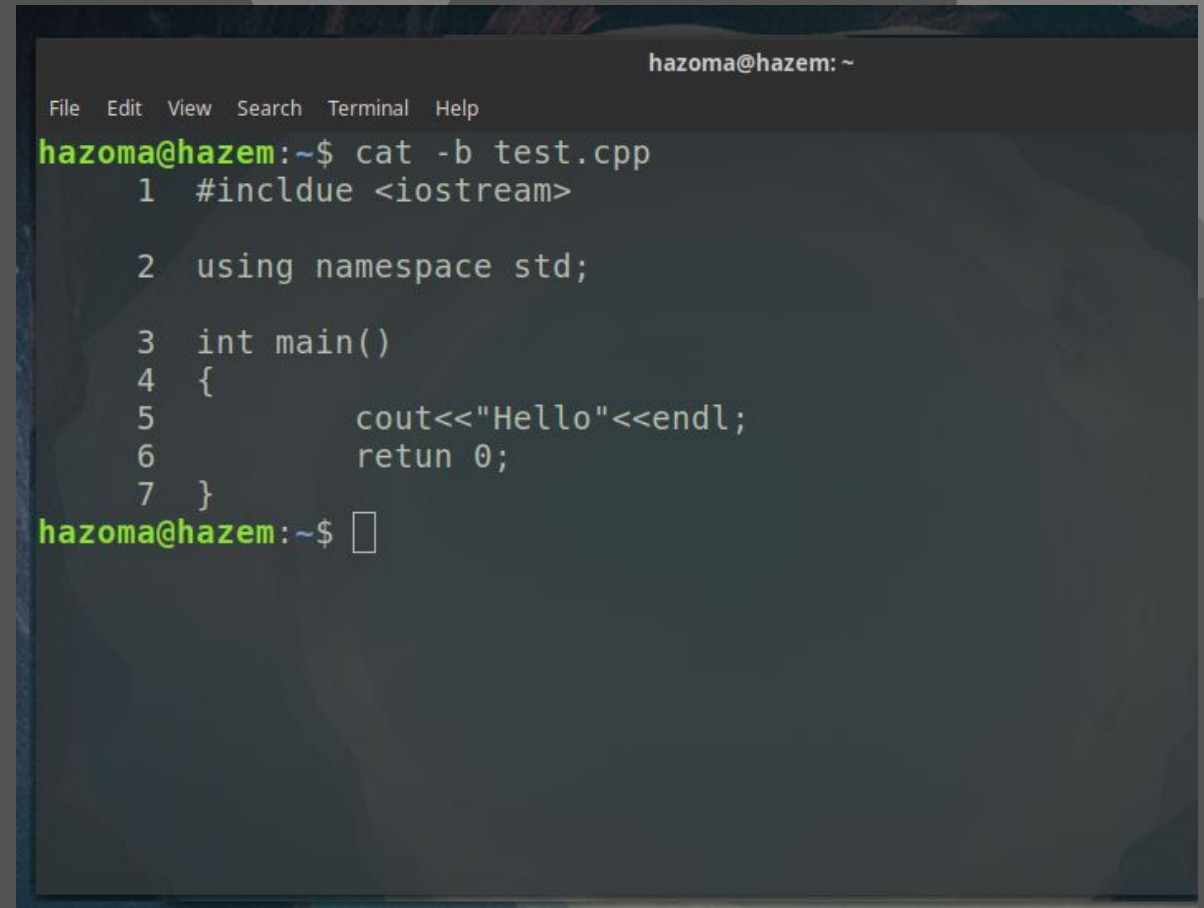
Cat

cat is a basic Linux command used to concatenate files and print output to the standard output.

We can use some options to format the output, we can use:

- n : to number the lines
 - b : to only number lines containing text
- like the example shown in the photo.

To print the content of more than one file (concatenate) using cat, just add more arguments to the command: `cat file1 file2 file3`



```
hazoma@hazem: ~  
File Edit View Search Terminal Help  
hazoma@hazem:~$ cat -b test.cpp  
 1  #include <iostream>  
  
 2  using namespace std;  
  
 3  int main()  
 4  {  
 5      cout<<"Hello"<<endl;  
 6      retun 0;  
 7  }  
hazoma@hazem:~$
```

grep

grep is used to search in files, we type grep in the terminal followed by the string/pattern/expression we want to find then the file we want to search in and we can also type multiple files to search in.

Example:

```
grep "Pattern" File1 /home/user/Desktop/File2
```

Commonly used options with grep:

- i : Ignores the case of the pattern we're searching for.
- c : Counts the number of lines which contain the pattern.
- v (invert) : Searches for the non matching lines.

```
hazoma@hazem:~$ grep m test.cpp
#include <iostream>
using namespace std;
int main()
hazoma@hazem:~$ |
```

Using g++ to compile C++ files

The background is a dark gray color. It features several abstract geometric shapes: several circles in various shades of gray (light, medium, and dark) and a few triangles in light gray. On the right side, there is a large, solid gold-colored circle. A dashed white line forms a curved arrow that starts near the bottom of this gold circle and points towards the top-left, ending near the top of the circle.

g++

G++ Compiler or GNU-C++ Compiler.

Install it by running: `sudo apt install g++`

By default an output file will be created by the name a.out which we will use to run our program.

To change the name of the output file we can use the option -o followed by the output file name then the c++ file. `g++ -o output main.cpp`

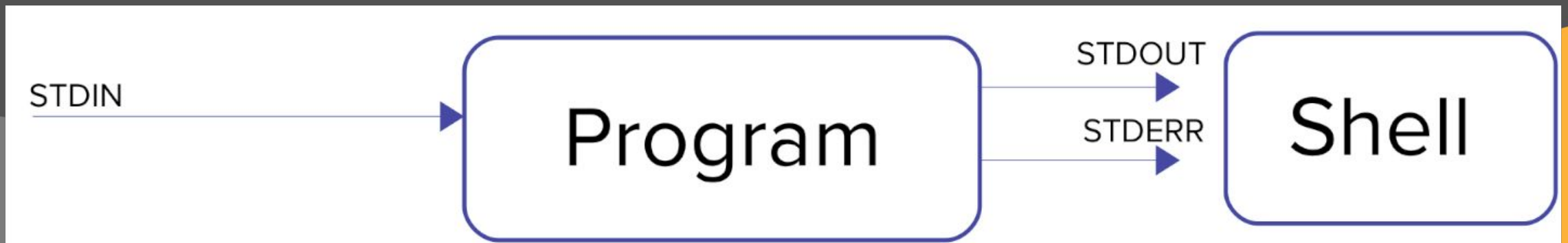
```
hazoma@hazem:~$ ls
test.cpp
hazoma@hazem:~$ g++ test.cpp
hazoma@hazem:~$ ls
a.out  test.cpp
hazoma@hazem:~$ ./a.out
Hello
hazoma@hazem:~$ |
```




Input and Output Streams and Redirection

Input and Output Streams

For every program, even operating systems, there is what we call standard input and output streams, they are used to take input from the user “standard input(stdin)”, and give the user output “standard output(stdout) and standard error(stderr)”.





Continuing..

These channels are the way to interact with any program.

Running `ls` gives normal output, thus channeled to stdout.

```
hazoma@hazem:~$ ls
a.out  screen.png  test.cpp
hazoma@hazem:~$ so
so: command not found
hazoma@hazem:~$ |
```

I/O Redirection

We can redirect stdout into file by using the greater than sign `>` or `>>` to append.

We can redirect stderr into file by using the sign `2>`.

We can also use `&>` to redirect both STDOUT and STDERR.

```
satharus@Argon: ~/Arduino$ ls
libraries
satharus@Argon: ~/Arduino$ ls > output.txt
satharus@Argon: ~/Arduino$ cat output.txt
libraries
output.txt
satharus@Argon: ~/Arduino$ ls
libraries  output.txt
satharus@Argon: ~/Arduino$ |
```

```
satharus@Argon: ~/Arduino$ hello
bash: hello: command not found
satharus@Argon: ~/Arduino$ hello > output2.txt
bash: hello: command not found
satharus@Argon: ~/Arduino$ cat output2.txt
satharus@Argon: ~/Arduino$ hello 2> err.txt
satharus@Argon: ~/Arduino$ cat err.txt
bash: hello: command not found
satharus@Argon: ~/Arduino$ |
```

Continuing..

You can also redirect input from a file into a program using the smaller than sign < followed by the filename.

```
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ cat test.cpp
#include <iostream>

using namespace std;

int main() {

    int x;
    cin >> x;
    cout << x * 4 << '\n';

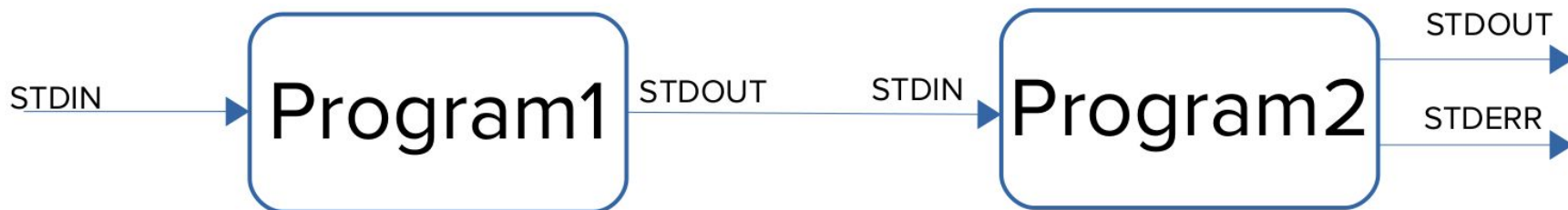
    return 0;
}
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ g++ test.cpp
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ cat file.txt
2
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ./a.out < file.txt
8
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ |
```

Piping ' | ':

What the pipe does is that it takes any output passed to it then uses it as an input for the following command.

We can use the pipe as many times as we want and to perform multiple commands sequentially with the I/O relations.

```
[ tawfik :: asus-laptop ]-( 0 )-[ ~ ]
$ ls -l
total 40
drwxrwxr-x 3 tawfik tawfik 4096 Jan  2 22:36 Android
drwxr-xr-x 2 tawfik tawfik 4096 Jan 23 21:36 Desktop
drwxrwxrwx 1 tawfik root   4096 Jan 19 16:39 Documents
drwxrwxrwx 1 root   root   4096 Jan 23 16:55 Downloads
drwxrwxrwx 1 tawfik root    0 Jul 22  2020 Music
drwxrwxrwx 1 root   root   4096 Jan  6 05:46 Pictures
drwxr-xr-x 2 tawfik tawfik 4096 Oct  3 02:48 Recordings
drwxr-xr-x 5 tawfik tawfik 4096 Jan 23 06:52 scripts
drwxr-xr-x 9 tawfik tawfik 4096 Jan 18 20:36 snap
drwxr-xr-x 2 tawfik tawfik 4096 Oct 25 14:52 Templates
drwxrwxrwx 1 tawfik root   4096 Dec 31 22:02 Videos
$ ls -l | grep Desktop
drwxr-xr-x 2 tawfik tawfik 4096 Jan 23 21:36 Desktop
```



File Permissions

The background is a dark gray field filled with various geometric shapes. On the left and center, there are several circles in different shades of gray, some with small triangles attached to their edges. On the right side, a large, solid gold circle is partially visible, with a dashed white arrow curving around its left edge, pointing towards the center of the slide.

Viewing Permissions:

The first column in `ls -l` command show us the file permissions

d

rwX

rwX

r-X

[TYPE][OWNER PERMISSIONS][GROUP PERMISSIONS][OTHER PERMISSIONS]

Type: This could be 'd' for directory, '-' for file, 'l' for link.

The three permissions are read 'r', write 'w', and execute 'x'.

```
[ tawfik :: asus-laptop ]-( 0 )-[ ~ ]
$ ls -l
total 40
drwxrwxr-x 3 tawfik tawfik 4096 Jan  2 22:36 Android
drwxr-xr-x 2 tawfik tawfik 4096 Jan 23 21:36 Desktop
drwxrwxrwx 1 tawfik root   4096 Jan 19 16:39 Documents
drwxrwxrwx 1 root  root   4096 Jan 23 16:55 Downloads
drwxrwxrwx 1 tawfik root    0 Jul 22  2020 Music
drwxrwxrwx 1 root  root   4096 Jan  6 05:46 Pictures
drwxr-xr-x 2 tawfik tawfik 4096 Oct  3 02:48 Recordings
drwxr-xr-x 5 tawfik tawfik 4096 Jan 23 06:52 scripts
drwxr-xr-x 9 tawfik tawfik 4096 Jan 18 20:36 snap
drwxr-xr-x 2 tawfik tawfik 4096 Oct 25 14:52 Templates
drwxrwxrwx 1 tawfik root   4096 Dec 31 22:02 Videos
```


Changing Permissions:

To change permissions we use the command `chmod`, and permissions are represented in 2 ways.

1 - Absolute “Numeric” Method:

We can use the command `chmod` followed by a 3 digit number then the file we want to change its permissions.

Each digit corresponds to a group of the three mentioned above, for each permission we have a number $x = 1$, $w = 2$, $r = 4$

r--	--x	-w-	rw-
100=3	001=1	010=2	110=6

Example: `chmod 764 file`

```
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
----r-xr-x 1 tawfik tawfik 0 Jan 24 02:38 file.txt
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ chmod 666 file.txt
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
-rw-rw-rw- 1 tawfik tawfik 0 Jan 24 02:38 file.txt
```

Symbolic Method:

Another way to change permissions is by referencing the group and the permissions.

u – Owner g – Group o – Others a – All users

Example: `chmod u+x file.txt`

```
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
-rw-rw-rw- 1 tawfik tawfik 0 Jan 24 02:38 file.txt
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ chmod u+x file.txt
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
-rwxrw-rw- 1 tawfik tawfik 0 Jan 24 02:38 file.txt
```

Changing Ownership:

The owner of the file is usually the user who created it or anyone who was given ownership of the file.

The third and fourth columns of the output of the `ls -l` command show the user owner and the group owner of the file respectively.

```
[ tawfik @ asus-laptop ]-( 0 )-[ /tmp/session ]  
$ ls -l  
total 0  
-rwxrw-rw- 1 tawfik tawfik 0 Jan 24 02:38 file.txt
```

Continuing..

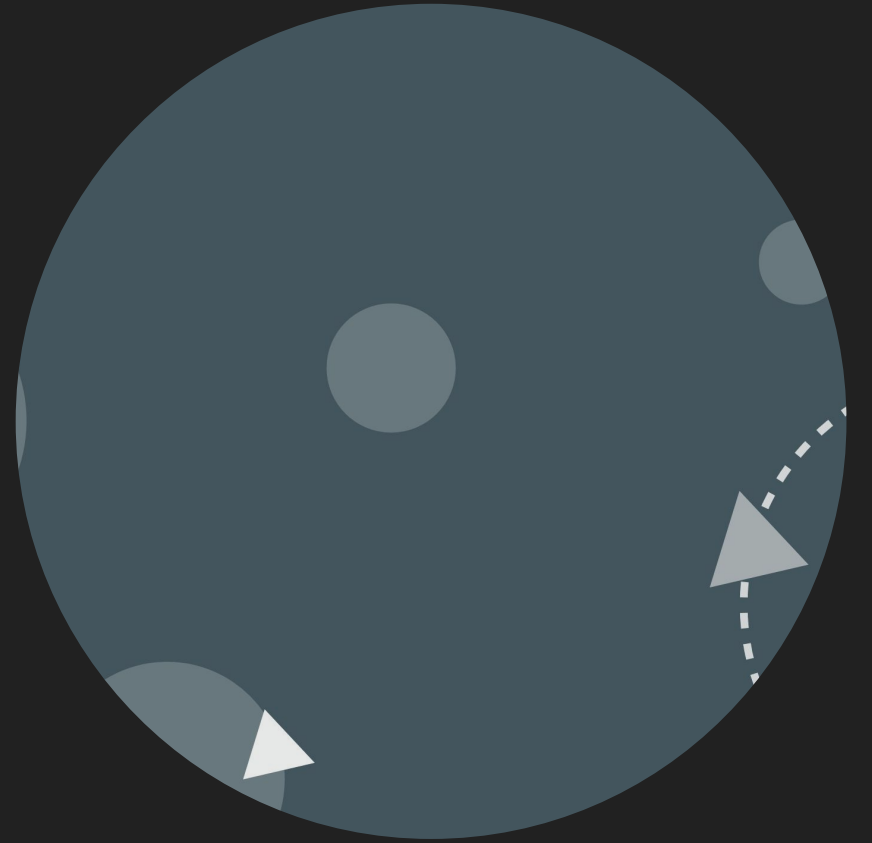
You can change the ownership of the file by using the command `chown` to change the user ownership and `chgrp` to change the group ownership.

Example: `chown tawfik file`

Example: `chgrp tawfik file`

```
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
-rw-rw-rw- 1 mostafa mostafa 0 Jan 24 02:50 file
-rw-rw-r-- 1 tawfik tawfik 0 Jan 24 02:51 file2
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ sudo chown tawfik file
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
-rw-rw-rw- 1 tawfik mostafa 0 Jan 24 02:50 file
-rw-rw-r-- 1 tawfik tawfik 0 Jan 24 02:51 file2
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ sudo chgrp tawfik file
[ tawfik :: asus-laptop ]-( 0 )-[ /tmp/session ]
$ ls -l
total 0
-rw-rw-rw- 1 tawfik tawfik 0 Jan 24 02:50 file
-rw-rw-r-- 1 tawfik tawfik 0 Jan 24 02:51 file2
```

Any Questions?



The background is a dark gray field decorated with several light gray circles of varying sizes and four bright orange triangles. The circles are positioned in the corners and center, while the triangles are placed near the edges. The text "Thank you" is centered in a white, cursive font.

Thank you