

Why maximus?

I decided to create this tool to use the very famous figlet (for more informations click here) tool in a customized way.

mini docs

First of all, open the terminal and write:

git clone https://github.com/AntonioBerna/maximus.git

now use the following command to access the project folder:

cd maximus

then just use the following command to build the project in your operating system in order to generate the executable file:

cargo build

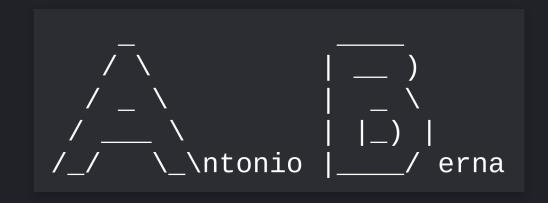
Once this procedure is finished you can use the program using the following command:

cargo run

In this way you will receive the following message which will help you understand how you should use the program, in particular:

Usage: target/debug/maximus <text>

For example, the command cargo run "AntonioBerna" produce the following result:



finally if you want to delete the executable in a simple way you can use the following command:

cargo clean

Ok... but how does it work?

First of all, you need to analyze the src/main.rs file and in particular the main function:

```
use std::env;

// ...

fn main() {
    let args: Vec<String> = env::args().collect();
    if args.len() != 2 {
        eprintln!("Usage: {} <text>", args[0]);
        std::process::exit(1);
    }

    let converter: FigletConverter = FigletConverter::new().expect("Failed to create FIGlet converter");
    let converted_text: String = converter.convert(&args[1]);

    println!("{}", converted_text);
}
```

Now we analyze the FigletConverter struct:

```
use figlet_rs::FIGfont;

struct FigletConverter {
    font: FIGfont,
}
```

But, for using the figlet-rs crate you need to add it to the Cargo.toml file:

```
# ...
[dependencies]
figlet-rs = "0.1.5"
```

Now we analyze the impl block with the new method:

and with the convert method:

```
impl FigletConverter {
    fn convert(&self, text: &str) -> String {
        let mut lines: Vec<String> = vec![
            String::new();
            self.font.convert("A").unwrap().to_string().lines().count()
        ];
        for ch in text.chars() {
            if ch.is_uppercase() {
                if let Some(rendered_char) = self.font.convert(&ch.to_string()) {
                    for (i, line) in rendered_char.to_string().lines().enumerate() {
                        lines[i].push_str(line);
            } else {
                for i in 0..lines.len() {
                    if i == lines.len() - 2 {
                        lines[i].push(ch);
                    } else {
                        lines[i].push(' ');
        lines.join("\n")
```

Thanks for your attention!