Rust 101, tutorial de Rust en español

Rust es un lenguaje imperativo, orientado a objetos y funcional, con gran soporte a concurrencia y de propósito general. En este tutorial de Rust daremos nuestros primeros pasos en Rust, un lenguaje que nace de las necesidades de tener un lenguaje que fuese rápido, concurrente y seguro. Aprenderemos a amar/odiar al estricto compilador y veremos algunas de las grandes ideas detrás de Rust. Sin embargo, esto requiere de una mayor atención por nuestra parte ya que Rust no es *otro lenguaje basado en C*. Algunas de sus características son:

- Abstracciones sin costo.
- Seguridad de memoria garantizada.
- Eliminación de las condiciones de carrera.
- Generalización basada en *traits.*
- Comparación de patrones.
- Inferencia de tipos.

Rust recoge influencias de muchos lenguajes, para dar lugar a un estilo único de programación que costará más de lo habitual aprender a usar de forma efectiva. Rust toma influencia de C++, OCaml, Haskell y en menor medida de Erlang, Alef, Limbo, Swift y otros lenguajes menos conocidos.

Índice del tutorial de Rust

13. Leer de teclado en Rust
14. Crear ventanas y botones en Rust con GTK
15. Leer y escribir JSON en Rust con Serde
16. Yew, crea webapps al estilo Angular/React en Rust
17. Construye un servidor web o una API con Rocket
18. Programa juegos en Rust con Piston (actualmente recomiendo ggez antes que Piston)
19. Futures y Tokio: programación asíncrona
20. Diversión con punteros: bloques unsafe en Rust
21. Bindings entre Rust y C/C++ con bindgen

Otros artículos sobre Rust en el blog

- Tutorial de Neon
- Tutorial de Maud
- WebAssembly y Rust
- Cheatsheet de contenedores en Rust

Rust funciona en una gran variedad de sistemas operativos y emite código compatible con muchos otros. El compilador oficial de Rust usa LLVM de forma interna, por lo que las mejoras que se realicen sobre LLVM mejoraran de forma indirecta el compilador de Rust. Otros lenguajes que usan LLVM son: C++ (con el compilador Clang), Julia, Swift, FORTRAN (con el compilador Flang), D (con el compilador LDC), Haskell (a través de GHC) y muchos más con otros compiladores. Mencionar que también existe un compilador de Rust bajo la suite GCC pero es experimental y no se recomienda su uso. Para instalar el compilador oficial de Rust visitamos la página oficial de Rust.

Rustup se trata de un programa que nos ayudará mucho y que nos permite instalar varias versiones de Rust en una misma máquina así como actualizar las mismas. Es una herramienta parecida a rvm de Ruby o nvm de Node.js. **rustup** nos instala **rustc**, **cargo** y **rustdoc**. Comprobemos que Rust se ha instalado correctamente, hagamos un Hola Mundo. Crea un archivo llamado *hola.rs* (rs es la extensión de los archivos de código fuente de Rust). Debe contener lo siguiente:

```
fn main() {

    println!("¡Hola mundo!");

}
```

Podemos compilar con **rustc**.

`rustc hola.rs -o hola.`

Como en GCC, la opción -o sirve para especificar el nombre del ejecutable de salida. **println!** es una macro que nos permite escribir en pantalla. Permite realizar interpolaciones, usando llaves. Por ejemplo: *println!("Mi nombres es {}","Adrián");*
Equivale a: *println!("Mi nombre es Adrián");*

Usando rustup.

He aquí algunos comandos útiles para manejar **rustup** de forma efectiva.

Mostrar que versión está actualmente activa.

`rustup show.`

Actualizar los entornos de Rust.

`rustup update.`

Instalar la versión nightly de Rust.

`rustup install nightly.`

Ejecutar esta un comando con un entorno determinado.

`rustup run nightly rustc hola.rs -o hola.`

Cambiar el entorno por defecto.

`rustup default nightly / rustup default stable.`

Añadir una plataforma al entorno (para cross-compiling)

`rustup target add arm-linux-androideabi` (Android ARM).