Getting Started with Rust – A Beginner's Guide

1. Title & Objective

Objective: To master the fundamentals of **Rust**, a systems programming language that guarantees memory safety and high performance.

- **Chosen Technology:** Rust.
- **Why Rust?** Unlike Python or Java, Rust offers "zero-cost abstractions" and handles memory without a garbage collector. It is currently the industry standard for performance-critical software (like browser engines and cloud infrastructure).
- **End Goal:** To set up a professional Rust environment, understand the Cargo ecosystem, and deploy a verified "Hello World" application.

---

2. Quick Summary of the Technology

What is it?

Rust is a multi-paradigm, high-level, general-purpose programming language designed for performance and safety, especially safe concurrency.

Where is it used?

- **Cloud Infrastructure:** Powering high-scale services at AWS and Google.
- **WebAssembly:** Running high-performance code in web browsers.
- **Operating Systems:** Significant portions of the Linux kernel and Windows are being rewritten in Rust.

Real-world Example: Discord switched their "Read States" service from Go to Rust, eliminating "stop-the-world" garbage collection spikes and drastically reducing latency.

---

3. System Requirements

- **OS:** Windows 10/11, macOS, or any modern Linux distribution (Ubuntu/Fedora).
- **Tools/Editors:** * **Visual Studio Code (VS Code)** is the recommended IDE.
    - **rust-analyzer extension** for VS Code (essential for real-time error checking).
- **Packages:** Build tools (linker) such as build-essential (Linux) or C++ Build Tools (Windows).

---

4. Installation & Setup Instructions

1. **Download Rustup:** Visit rustup.rs and copy the installation command.

Run in Terminal:
Bash
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
2.

Configure Environment: Follow the on-screen prompts to proceed with the "Default" installation. Once finished, restart your terminal or run:
Bash
source $HOME/.cargo/env
3.

Verify: Check the versions of the compiler and package manager.
Bash
rustc --version
cargo --version
4.

---

5. Minimal Working Example

This example initializes a new project and prints a message to the console using the Rust macro system.

Code (src/main.rs):

Rust
```
fn main() {
    // println! is a macro (denoted by !) that prints text to the console.
    // Unlike functions, macros can take a variable number of arguments.
    println!("Hello, World! Welcome to the world of Rust 🦀");
}
```

Expected Output:

Plaintext
```
  Compiling hello_rust v0.1.0
    Finished dev [unoptimized + debuginfo] target(s) in 0.45s
     Running `target/debug/hello_rust`
Hello, World! Welcome to the world of Rust 🦀
```

---

6. AI Prompt Journal

Prompt Category	Prompt Used	Learning Reflection
Explanation	"Explain the difference between a function and a macro in Rust using the println! example."	I learned that macros generate code at compile-time, which is why println! can handle different types and numbers of arguments safely.
Documentation	"Create comprehensive Rustdoc comments for a main function that prints a greeting."	Learned that /// is the standard for documentation and that Rustdoc supports Markdown and example code testing.
Comparison	"Why is Rust considered safer than C++ even though it doesn't use a garbage collector?"	The AI explained the **Ownership** and **Borrowing** rules, which prevent memory leaks at compile-time rather than run-time.

---

7. Common Issues & Fixes

- **Error:** linker 'cc' not found
  - **Fix:** Your system lacks a C linker. Run sudo apt install build-essential (Ubuntu) or install Xcode Command Line Tools (macOS).
- **Error:** cargo: command not found
  - **Fix:** The path isn't updated. Run source $HOME/.cargo/env or restart your computer.
- **Error:** expected ';', found '}'
  - **Fix:** Rust is expression-based and very strict with semicolons. Ensure every statement ends with ; unless it is a returned expression.

---

8. References

- **Official Docs:** The Rust Programming Language (The Book)
- **Quick Start:** Rust-lang Getting Started
- **Video Tutorial:** Rust in 100 Seconds (Fireship)