#System Programming in Rust: Beyond Safety#

Why today security has become an issue?

In 2017, the common vulnerabilities and exposure database lists that two-thirds of the total exploits were result of unsafe programming language principles used by the humans such as, complicated details of object lifetimes, bounds checking,pervasive use of pointer aliasing, pointer arithmetic, and unsafe type casts.

Why people still use C?

The main traditional usage is the performance.

How safe is RUST?

Rust enforces type and memory safety through a restricted ownership model, where there exists a unique reference to each live object in memory. This allows statically tracking the lifetime of the object and de allocating it without a garbage collector. The runtime over- head of the language is limited to array bounds checking, which is avoided in most cases by using iterators.

What is the specialty of RUST in comparing to other programming languages?

Rust’s linear type system enables capabilities missing in traditional programming languages (both safe and unsafe). This paper identify three categories of such capabilities: isolation, analysis, and automation.

What is a system programming language ?

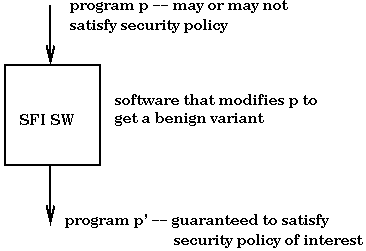
A system programming language is a programming language used for system programming; such languages are designed for writing system software, which usually requires different development approaches when compared with application software.

System software is computer software designed to operate and control the computer hardware, and to provide a platform for running application software. System software includes software categories such as operating systems, utility software, device drivers, compilers, and linkers.

What is Software Fault Isolation?

We have been discussing protection measures that a single operating system can provide. One way to think of this is to view the operating system as a padded cell in which programs operate; we have been discussing the nature of the padding. Another way to get programs to behave in a manner consistent with a given security policy is by "brainwashing." That is, modify the programs so that they behave only in safe ways. This is embodied by a recent approach to security known as *software-based fault isolation* (SFI).

So far, the environment has been responsible for policy enforcement, where the environment is either the OS/kernel or the hardware. Hardware methods include addressing mechanisms (e.g. virtual memory); OS methods include having two modes (where the supervisor mode has access to everything). The new approach we discuss today is to construct a piece of software that transforms a given program p into a program p', where p' is guaranteed to satisfy a security policy of interest.



Is RUST similar to SFI?

Existing SFI implementations do not support sending data across protection boundaries by reference, as this enables the sender to maintain access to the data. Hence a copy is required to ensure isolation, making such solutions unacceptable in line-rate systems. Rust’s single ownership model allows us to implement zero-copy SFI.

What is static information flow control (IFC)

It enforces confidentiality by tracking the propagation of sensitive data through the program. But it is complicated when having aliasing. in the IFC case, writing sensitive data to an object makes this data accessible via all aliases to the object and can therefore change the security state of multiple variables. By restricting aliasing, Rust sidesteps the problem