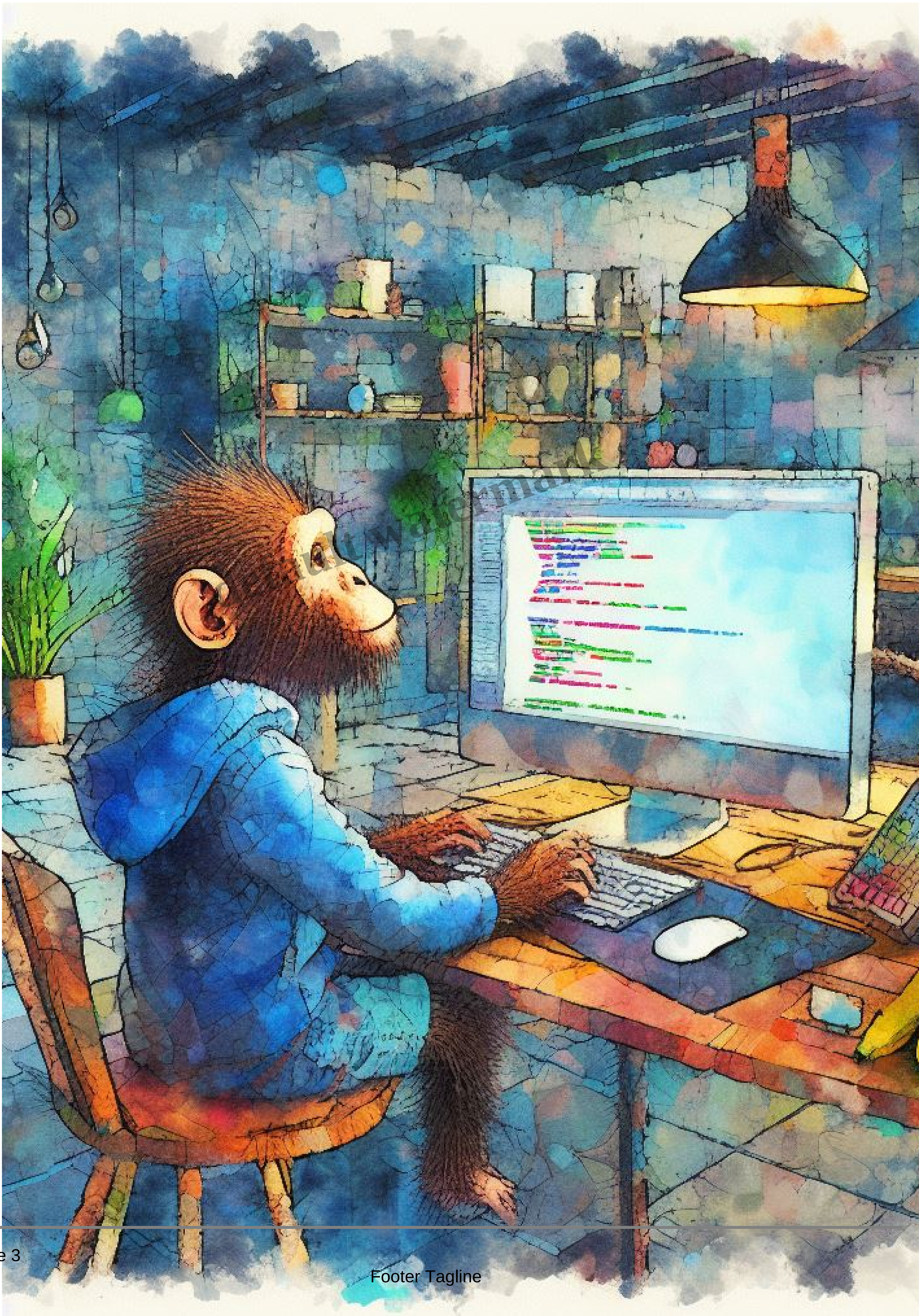
What is the difference between a raw pointer and a function pointer in Rust?

Description



A **raw pointer** and a **function pointer** are both types of pointers in programming, but they have different purposes and use cases:

Raw Pointer:

- A raw pointer is a simple, low-level pointer that stores the memory address of an object or a location in memory.
- Raw pointers are typically used for low-level memory manipulation and interaction with foreign functions or libraries, such as when interfacing with C code.
- Raw pointers in Rust are of two types: mutable (`*mut T`) and immutable (`*const T`), where `T` is the type of data they point to.
- Raw pointers do not carry information about the type of data they point to, and they don’t have any associated safety guarantees. It’s the programmer’s responsibility to ensure that they are used safely.

Function Pointer:

- A function pointer, as the name suggests, is a pointer that stores the memory address of a function.
- Function pointers allow you to treat functions as first-class citizens, meaning you can pass functions as arguments, return functions from other functions, and store functions in data structures.
- In Rust, function pointers have a specific type signature that describes the function’s parameter types and return type. For example, `fn(i32, i32) -> i32` is a function pointer type that points to a function taking two `i32` arguments and returning an `i32`.
- Function pointers in Rust are typically used for scenarios where you need to dynamically select or call different functions based on some condition.

In summary, the key differences are that raw pointers are used for general memory address manipulation and don’t carry type information, while function pointers specifically point to functions and include type information about the function’s signature. Function pointers are used for dynamic dispatch and function selection in Rust, and they are a higher-level abstraction compared to raw pointers.

How do their sizes compare ?

The size of **raw pointers** and **function pointers** can vary depending on the architecture and compiler, but there are some general principles to consider:

1. Raw Pointers:

- The size of a raw pointer depends on the target architecture.
- On a 32-bit architecture, a raw pointer typically takes 4 bytes (32 bits) of memory because it needs to store a 32-bit memory address.
- On a 64-bit architecture, a raw pointer typically takes 8 bytes (64 bits) of memory because it needs to store a 64-bit memory address.

1. Function Pointers:

- The size of a function pointer depends on the target architecture and the function's signature (parameter types and return type).
- In Rust, the size of a function pointer is not fixed but is determined by the ABI (Application Binary Interface) rules for the specific function signature. Different function signatures may have different pointer sizes.
- For example, a function pointer with the signature `fn(i32) -> i32` may have a different size than a function pointer with the signature `fn(f64) -> f64`.
- Function pointers often include additional information about the function's calling convention, which can affect their size.

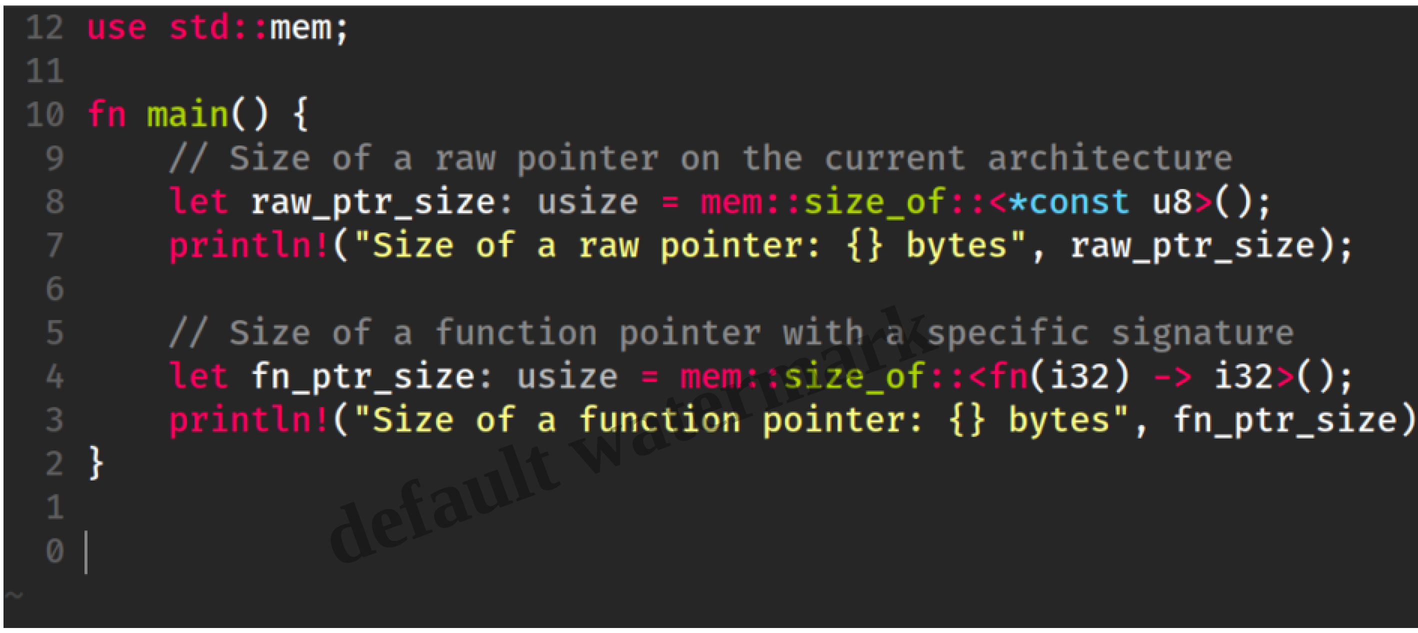
To determine the size of a specific pointer type in Rust, you can use the `std::mem::size_of` function. For example:

```
use std::mem;

fn main() {
    // Size of a raw pointer on the current architecture
    let raw_ptr_size = mem::size_of::<*const u8>();
    println!("Size of a raw pointer: {} bytes", raw_ptr_size);

    // Size of a function pointer with a specific signature
    let fn_ptr_size = mem::size_of::<fn(i32) -> i32>();
    println!("Size of a function pointer: {} bytes", fn_ptr_size);
}
```

This code will print the size of both raw pointers and function pointers on your target architecture. Keep in mind that the actual sizes may vary depending on the platform and compiler you are using.

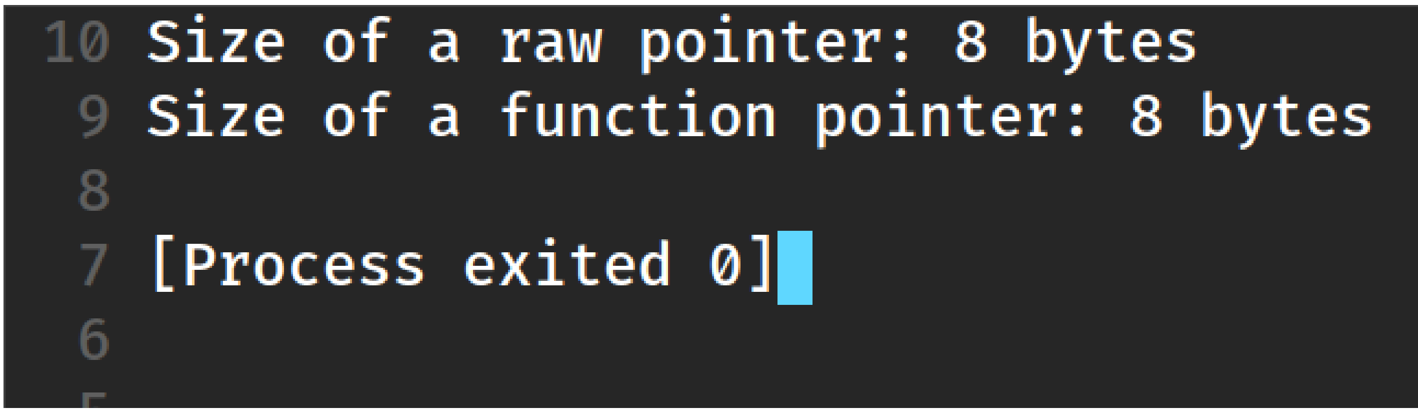


```
use std::mem;

fn main() {
    // Size of a raw pointer on the current architecture
    let raw_ptr_size: usize = mem::size_of::<*const u8>();
    println!("Size of a raw pointer: {} bytes", raw_ptr_size);

    // Size of a function pointer with a specific signature
    let fn_ptr_size: usize = mem::size_of::<fn(i32) -> i32>();
    println!("Size of a function pointer: {} bytes", fn_ptr_size)
}
```

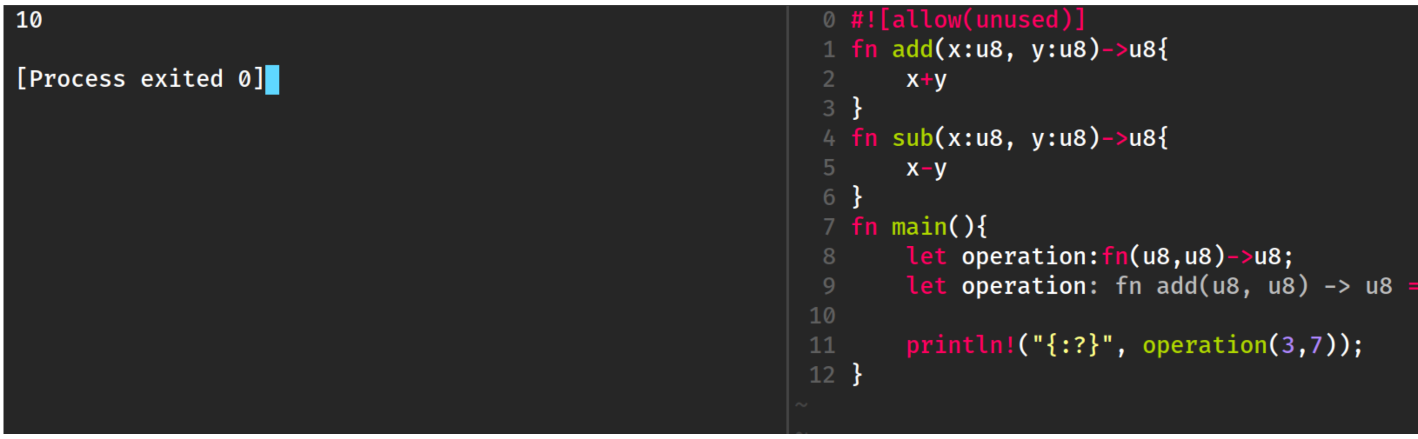
```
x86_64 x86_64 x86_64 GNU/Linux
```



```
Size of a raw pointer: 8 bytes
Size of a function pointer: 8 bytes

[Process exited 0]
```

Pointer to a function



```
10

[Process exited 0]
```

```
#![allow(unused)]
fn add(x:u8, y:u8)->u8{
    x+y
}
fn sub(x:u8, y:u8)->u8{
    x-y
}
fn main(){
    let operation:fn(u8,u8)->u8;
    let operation: fn add(u8, u8) -> u8 =

    println!("{:?}", operation(3,7));
}
```

Link to Rust playground :

https://play.rust-lang.org/?version=stable&mode=debug&edition=2021&gist=cd0969c745ec9edc01420b523b1040a8

Links

https://web.mit.edu/rust-lang_v1.25/arch/amd64_ubuntu1404/share/doc/rust/html/reference/types.html#function-pointer-types

Category

1. Rust Programming

Tags

1. Rust Programming

Date Created
2023/10/08
Author
admin