Defect
Detect

DUmps Binaries Logs INternals
School of Security

# Rust

## Memory Thinking

Dmitry Vostokov
Software Diagnostics Services

# Memory Thinking for Rust

Slides with Descriptions and Source Code Illustrations

**Dmitry Vostokov**
**Software Diagnostics Services**

OpenTask

Memory Thinking for Rust: Slides with Descriptions and Source Code Illustrations

Published by OpenTask, Republic of Ireland

Copyright © 2024 by OpenTask

Copyright © 2024 by Dmitry Vostokov

Copyright © 2024 by Software Diagnostics Services

Copyright © 2024 by Dublin School of Security

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, transmitted in any form or by any means, or used for training artificial intelligence systems without the prior written permission of the publisher.

OpenTask books are available through booksellers and distributors worldwide. For further information or comments, send requests to press@opentask.com.

Product and company names mentioned in this book may be trademarks of their owners.

A CIP catalog record for this book is available from the British Library.

ISBN-13: 978-1912636662 (Paperback)

Revision 1.00 (June 2024)

# Table of Contents

**Table of Contents** 3

**Preface** 15

**About the Author** 16

**Introduction** 17

*Prerequisites* 18

*Training Goals* 19

*Warning* 20

*Training Principles* 21

*Schedule* 22

*Training Idea* 23

*General Rust Aspects* 24

*What We Do Not Cover* 26

*Linux Rust Aspects* 27

*Windows Rust Aspects* 28

*Why Rust?* 29

*My Genealogy of Rust* 31

*Rust Mastery Process* 32

*Thought Process* 33

**Philosophy of Unsafe Pointers** **34**

*A General Pointer Concept* *35*

*Pointer* *36*

*Pointer Dereference* *37*

*Many to One* *38*

*Many to One Dereference* *39*

*Invalid Pointer* *40*

*Invalid Pointer Dereference* *41*

*Wild (Dangling) Pointer* *42*

*Pointer to Pointer* *43*

*Pointer to Pointer Dereference* *44*

*Naming Pointers and Entities* *45*

*Names as Pointer Content* *46*

*Pointers as Entities* *47*

**Unsafe Rust Code Examples** **48**

*Unsafe Pointer* *49*

*Unsafe Pointer Dereference* *51*

*Unsafe Many to One* *53*

*Unsafe Many to One Dereference* *55*

*Invalid Pointer* *57*

*Invalid Pointer Dereference (Alignment)* *58*

*Invalid Pointer Dereference (Access Violation)* *59*

*Wild (Dangling) Pointer* *60*

*Unsafe Pointer to Pointer* *62*

*Unsafe Pointer to Pointer Dereference* *64*

**Philosophy of Values** **66**

*Values and Owners* *67*

*Moving Values* *69*

*Copying Values* *71*

*Dropping Values* *73*

*Ownership Tree* *75*

*Ownership Tree and Drops* *77*

*Partial Drops* *81*

*Multiple Owners (not in Rust)* *82*

*Multiple Owners and Drops* *83*

*Owners vs. Pointers* *84*

**Rust: A Copernican Revolution** **85**

*Values Revolve around Pointers* *86*

*Owners Revolve around Values* *87*

**Rust Philosophy of Values** **88**

*Restricted Ownership* *89*

*Value Lifetime* *91*

*Owner Lifetime* *94*

**Rust Philosophy of Pointers** **96**

*Types of Pointers* *97*

*References as Pointer Types* *98*

*References as Addresses* *100*

*Borrowing References* *103*

*Reference Lifetime* *105*

**x64 Disassembly Review (WinDbg)** **107**

*x64 CPU Registers* *108*

*Instructions and Registers* *109*

*Memory and Stack Addressing* *110*

*Memory Cell Sizes* *111*

*Memory Load Instructions* *112*

*Memory Store Instructions* *113*

*Flow Instructions* *114*

*Function Parameters* *115*

*Struct Function Parameters* *116*

**x64 Disassembly Review (GDB AT&T Flavor)** **117**

*x64 CPU Registers* *118*

*x64 Instructions and Registers* *119*

*Memory and Stack Addressing* *120*

*x64 Memory Load Instructions* *121*

*x64 Memory Store Instructions* *122*

*x64 Flow Instructions* *123*

*x64 Function Parameters* *124*

*x64 Struct Function Parameters* *125*

**ARM64 Disassembly Review** **126**

*A64 CPU Registers* *127*

*A64 Instructions and Registers* *128*

*Memory and Stack Addressing* *129*

*A64 Memory Load Instructions* *130*

*A64 Memory Store Instructions* *131*

*A64 Flow Instructions* *132*

*A64 Function Parameters* *133*

*A64 Struct Function Parameters* *134*

**Memory Storage** **135**

*Memory Regions* *136*

*Dynamic Virtual Memory* *137*

*Static Memory* *138*

*Rust Static Memory Values* *139*

*Rust Static Memory References* *140*

*Stack Memory* *145*

*Thread Stack Frames* *146*

*Local Value Lifecycle* *147*

*Rust Stack Memory Values* *148*

*Rust Stack Memory References* *149*

*Heap Memory* *152*

*Rust Heap Memory Values* *153*

*Useful WinDbg Commands* *156*

*Useful GDB Commands* *157*

**Memory and Pointers** **158**

*Mental Exercise* *159*

*Debugger Memory Layout* *160*

*Memory Dereference Layout* *161*

*Names as Addresses* *162*

*Addresses and Entities* *163*

*Addresses and Structures* *164*

*Pointers to Structures* *165*

*Arrays* *166*

*Arrays and Pointers to Arrays* *167*

*Fat Pointers* *168*

*Array Slices* *169*

*String Literals (UTF-8)* *173*

*Byte Strings* *176*

*Vectors* *178*

*Vector Slices* *179*

*Strings* *183*

*String Slices* *184*

*C-Strings* *187*

*C-String Slices* *188*

**Basic Types** **191**

*Bytes, References, and Pointers* *192*

*u32, Pointers, and References* *195*

*Little-Endian System* *198*

*u64 and Pointers* *200*

*Size* *203*

*Alignment* *209*

**Entity Conversion** **211**

*Conversion through Pointers* *212*

*Safe Conversion* *215*

*Forcing* *217*

**Tuples** **219**

**Structs** **222**

*Tuple-like Structs* *223*

*Newtypes* *227*

*Named-field Structs* *230*

*Reference/Pointer to Struct* *235*

*Ref/Ptr to Struct Dereference* *236*

*Dereference with Replacement* *237*

*Many Ref/Ptr to One Struct* *240*

*Many to One Dereference* *241*

*Ref/Ptr to Ref/Ptr to Struct* *242*

*Ref/Ptr to Ref/Ptr Dereference* *243*

**Memory and Structs** **248**

*Addresses and Structs* *249*

*Struct Field Addresses* *250*

*Ref/Ptr to Structs* *254*

*Ref/Ptr to Struct and Fields* *255*

*External Struct Alignment* *259*

*Internal Struct Alignment (WinDbg)* *260*

*Internal Struct Alignment (GDB)* *261*

**Source Code and Symbols** **264**

*Conceptual Layer (Modules)* *265*

*Logical Layer (Crates)* *266*

*Physical Layer (Source Files)* *267*

*Name Isolation* *268*

**Functions** **271**

*Pointers to Functions* *272*

*References to Functions* *275*

*Function Pointer Types* *277*

*Struct Function Fields* *278*

*Associated Functions* *280*

*Pointers to Associated Functions* *282*

*Type-associated Functions* *284*

*Trait Functions* *286*

*Trait Objects* *287*

*vtable Memory Layout* *289*

*Trait Object Memory Layout* *290*

*Boxed Trait Object Layout* *296*

*Struct Constructors* *300*

*Struct Destructor* *303*

*Struct Clone* *308*

*Struct Copy* *310*

*Parameters by Value* *312*

*Parameters by Ref/Ptr* *316*

*self* *319*

**Closures and Captures** **321**

*Closure Struct* *322*

*A64 Closure Struct Example* *324*

*Captures (Borrowing)* *325*

*Captures (Borrowing) x64 Linux* *326*

*Captures (Move)* *328*

*Captures (Move) x64 Linux* *329*

**Pinning** **331**

*Use Cases* *332*

**Rust Books** **338**