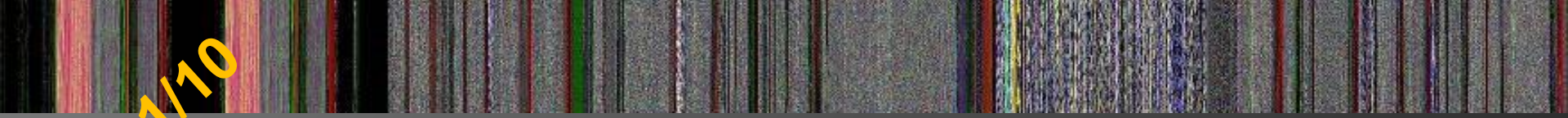


Part 1/10

C & C++

Linux Diagnostics

Accelerated

Dmitry Vostokov
Software Diagnostics Services

Prerequisites

- Development experience

and (optional)

- Basic core dump analysis

Training Goals

- Review common fundamentals of C and C++
- Review C++ specifics
- Use GDB for learning C and C++ internals
- See how C and C++ knowledge is used during diagnostics and debugging

Training Principles

- Talk only about what I can show
- Lots of pictures
- Lots of examples
- Original content and examples

Schedule

- `std::vector<Session> sessions;`
- `assert(sessions.size() == 5);`
- `assert(sessions.capacity() > 5);`

Training Idea

- Similar course for Windows
- Core dump analysis training
- Reversing training
- Linux API training

General C & C++ Aspects

- Philosophy of pointers
- Structures, classes, and objects
- Promotions and conversions
- Macros, types, and synonyms
- Source code organization, PImpl
- Pointer dereference walkthrough
- Functions and function pointers
- Inheritance
- Operators, function objects
- Destructors, virtual destructors
- Local stack variables and values
- Memory operators and expressions
- Alignment
- Slicing
- Iterators as pointers
- Lambdas and their internals
- Threads and synchronization
- Memory and pointers
- Basic types
- Memory and structures
- Uniform initialization
- Memory storage
- References
- Values, lvalues, rvalues
- Constant values and expressions
- Namespaces
- Constructors, copy, assignment
- Virtual functions, pure methods
- VTBL and VPTR
- Access levels
- Overloading, overriding
- Templates
- Memory ownership, RAII
- Smart pointers

What We Do Not Cover*

- Enumerations
- Move constructors and assignment operators
- Deleted and default members
- Universal references
- Concepts
- Coroutines
- Modules
- Tasks
- Ranges
- Container and algorithm semantics and pragmatics
- Container allocators
- Polymorphic allocators

* We promise to include these topics in the second edition

Linux C & C++ Aspects

- Linux-specific type aliases and macros
- LP64
- Necessary x64 and A64 disassembly
- Parameter passing
- Implicit parameter

Why C & C++?

- Interfacing
- Malware analysis
- Vulnerability analysis and exploitation
- Reversing
- Diagnostics
- Low-level debugging
- OS Monitoring
- Memory forensics
- Crash and hang analysis
- Secure coding
- Static code analysis
- Trace and log analysis

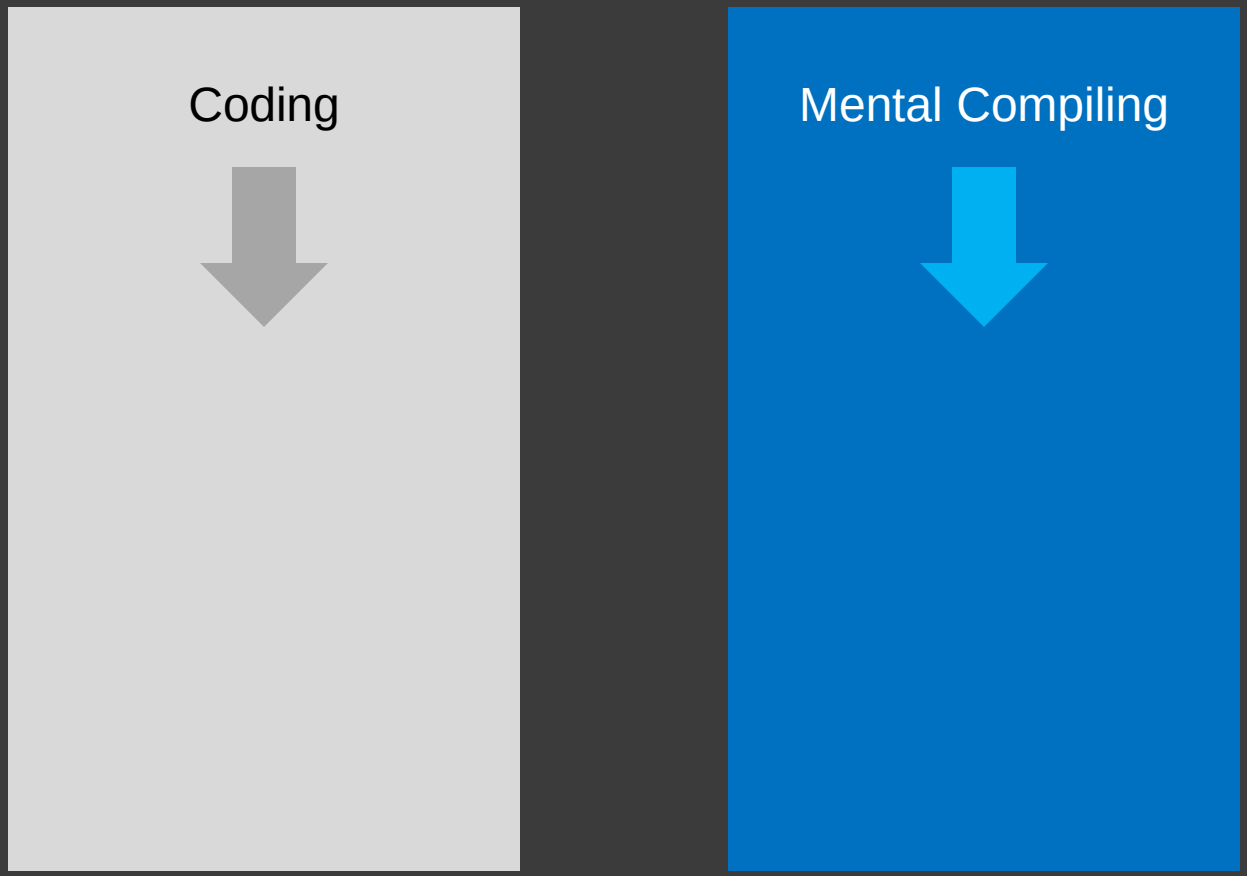
Which C & C++?

- C
- C++ as a better C
- Proper C++ (legacy and modern)
- Linux specifics

My History of C & C++

- C from 1987 and C++ from 1989 (Old CV)
- C++ as a better C from 1991
- Implicit design patterns in 1994-1995
- C++ as proper C++ from 2000
- Explicit design patterns in 2000
- C++98/03/STL from 2001
- BSD core dump analysis from 2012
- Linux core dump analysis from 2015
- […]
- C++11/14 from 2016
- C++17 from 2017
- Functional programming from 2020
- Linux system programming since 2022
- C++20 from 2023

C and C++ Mastery Process



Thought Process

- C and C++
- Scala/FP
- Python

Memory

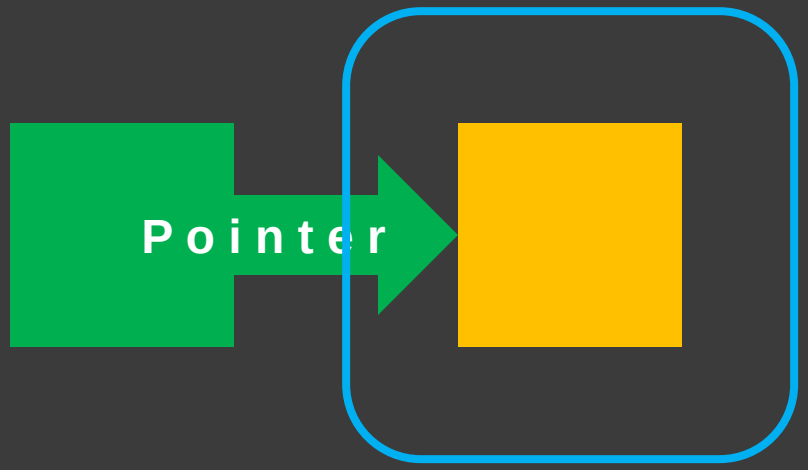
Functions

Data

Philosophy of Pointers

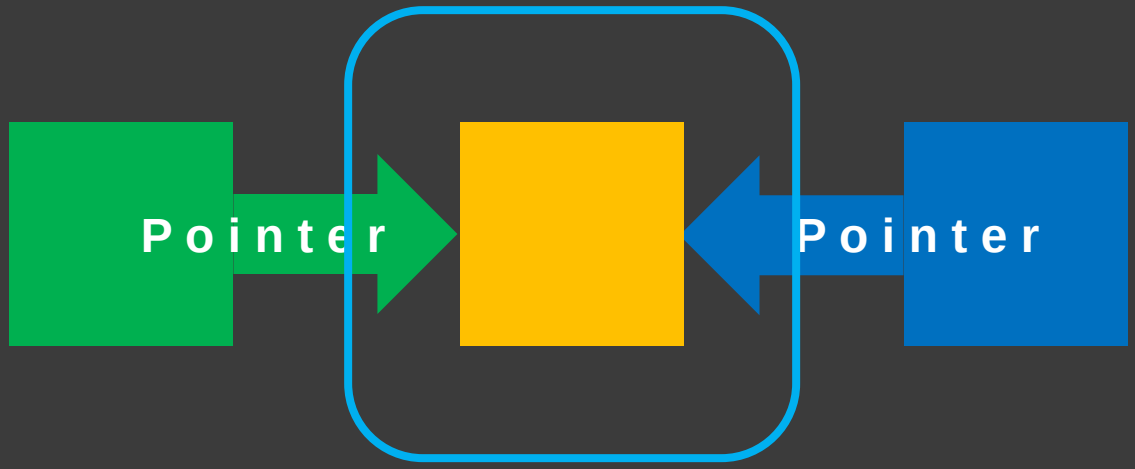
Pointer

Pointer Dereference

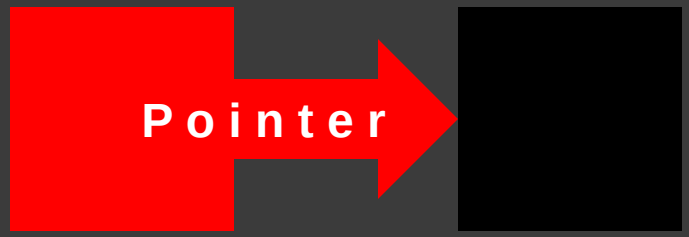


Many to One

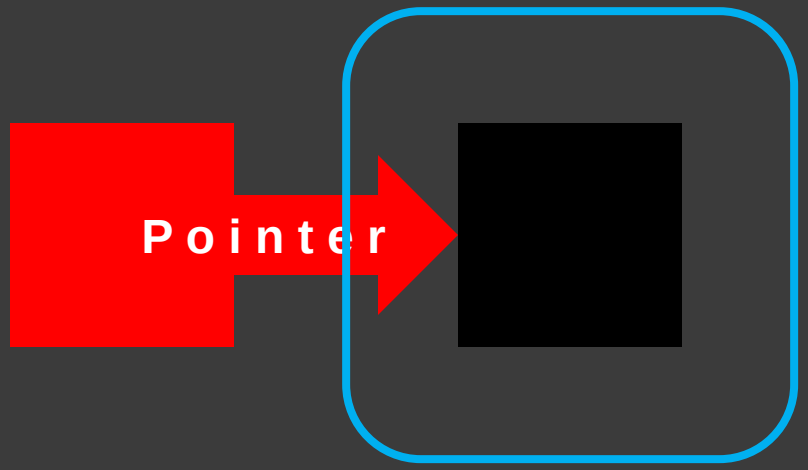
Many to One Dereference



Invalid Pointer



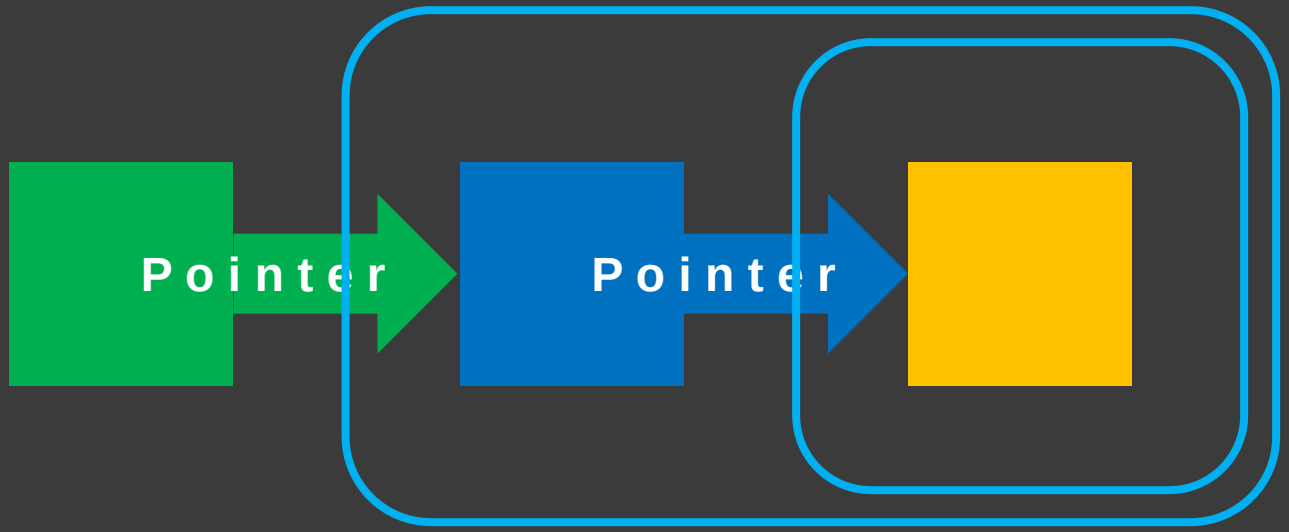
Invalid Pointer Dereference



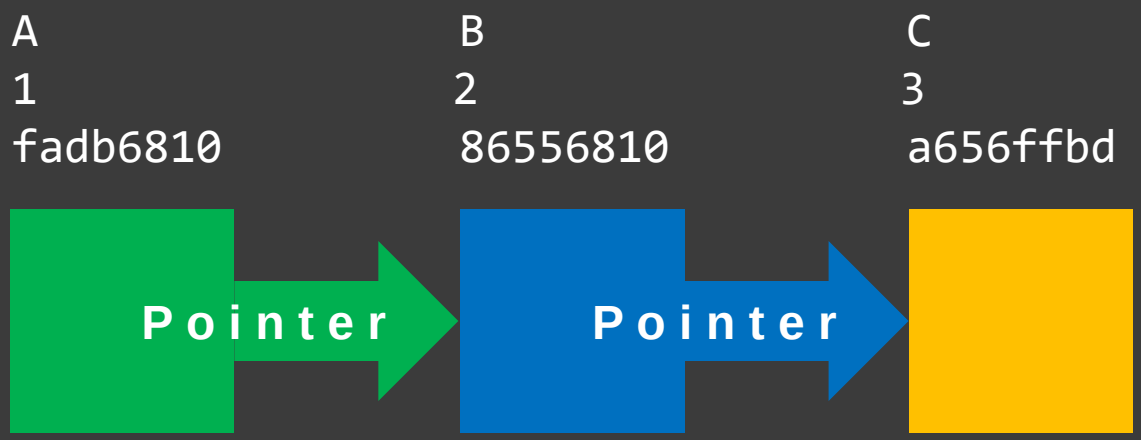
Wild (Dangling) Pointer

Pointer to Pointer

Pointer to Pointer Dereference

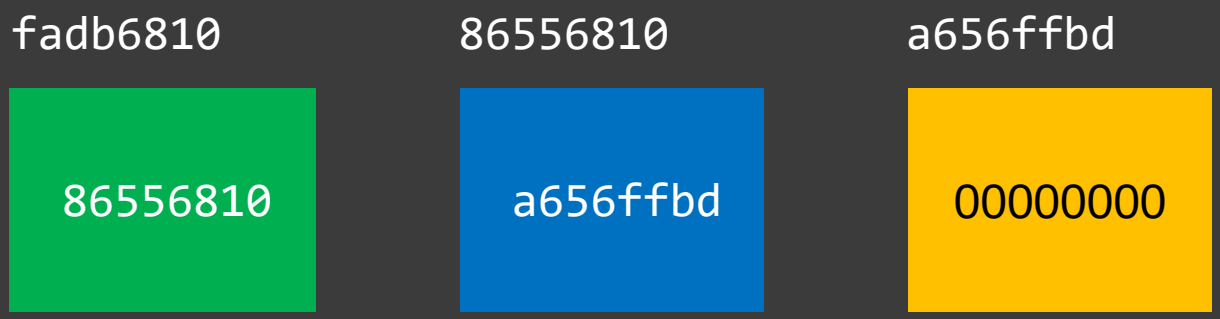


Naming Pointers and Entities



Names as Pointer Content

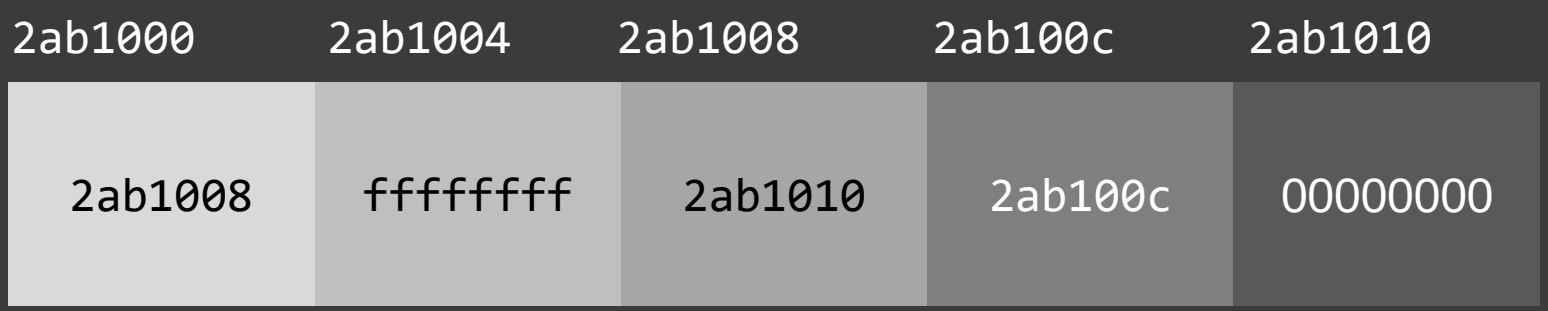
Pointers as Entities



Memory and Pointers

Mental Exercise

How many pointers can you count?



Debugger Memory Layout

2ab1000:	2ab1008
2ab1004:	ffffffff
2ab1008:	2ab1010
2ab100c:	2ab100c
2ab1010:	00000000
2ab1014:	00002000

2ab1000:	2ab1008	ffffffff
2ab1008:	2ab1010	2ab100c
2ab1010:	00000000	00002000

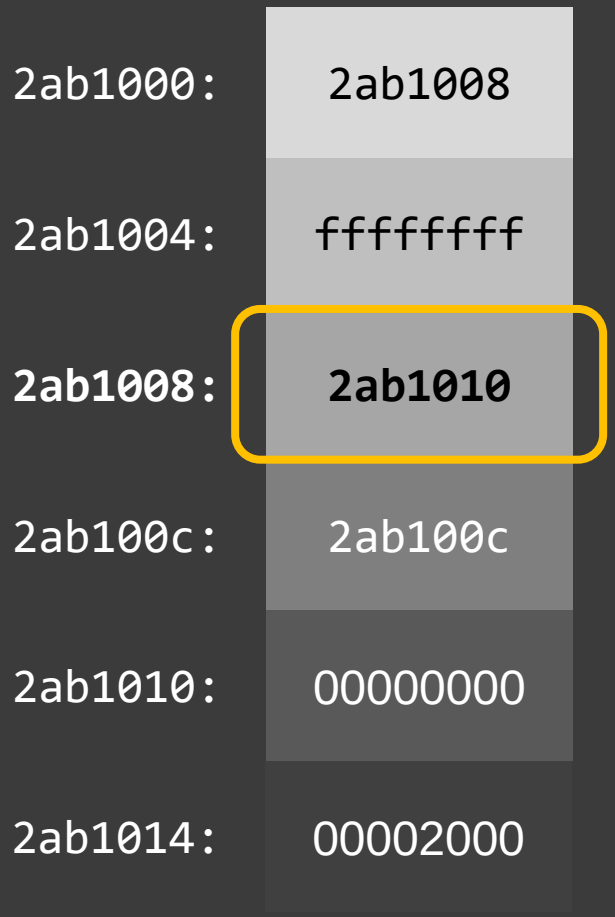
Memory Dereference Layout

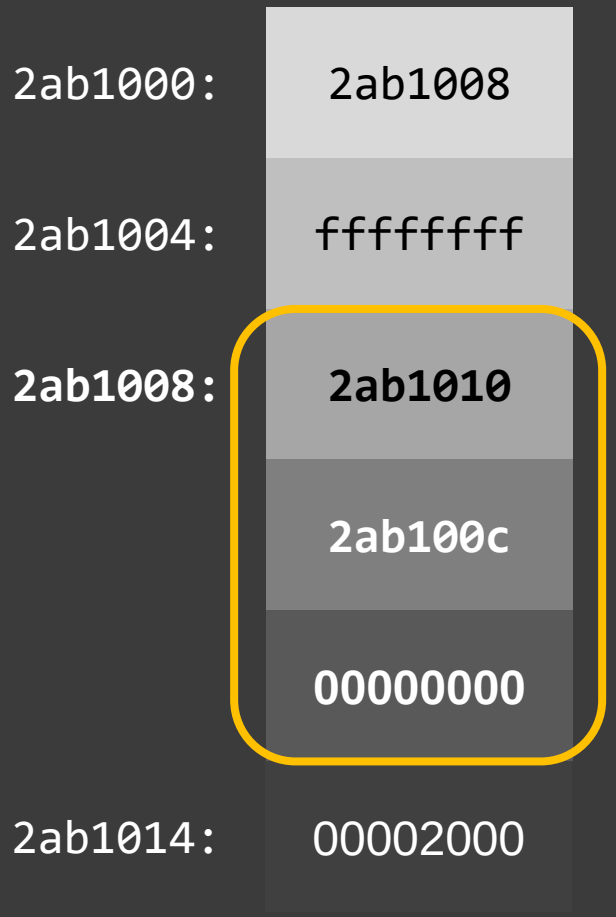
2ab1000:	2ab1008
2ab1004:	ffffffff
2ab1008:	2ab1010
2ab100c:	2ab100c
2ab1010:	00000000
2ab1014:	00002000

2ab1000:	2ab1008	2ab1010
2ab1004:	ffffffff	????????
2ab1008:	2ab1010	00000000
2ab100c:	2ab100c	2ab100c
2ab1010:	00000000	????????
2ab1014:	00002000	????????

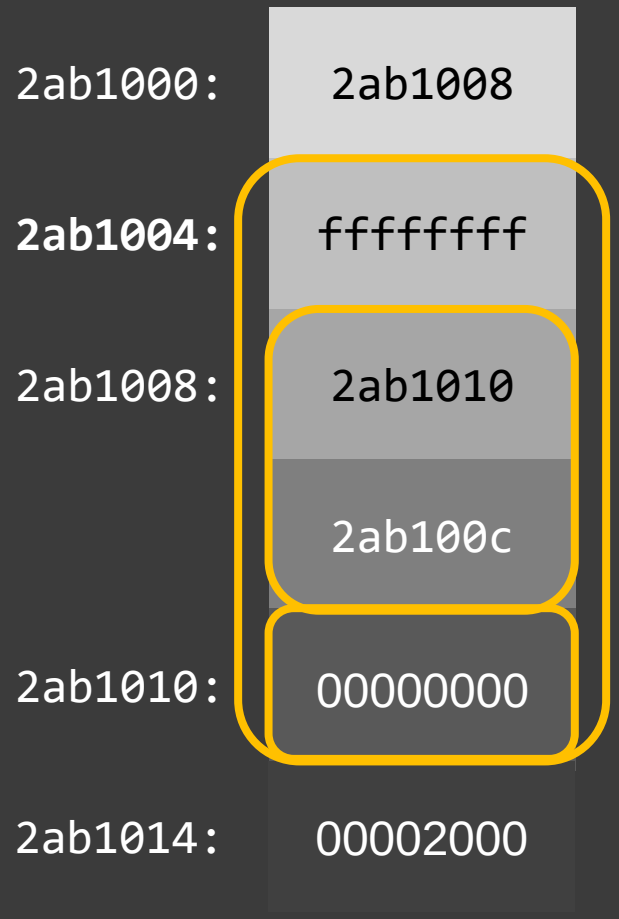
Names as Addresses

Addresses and Entities

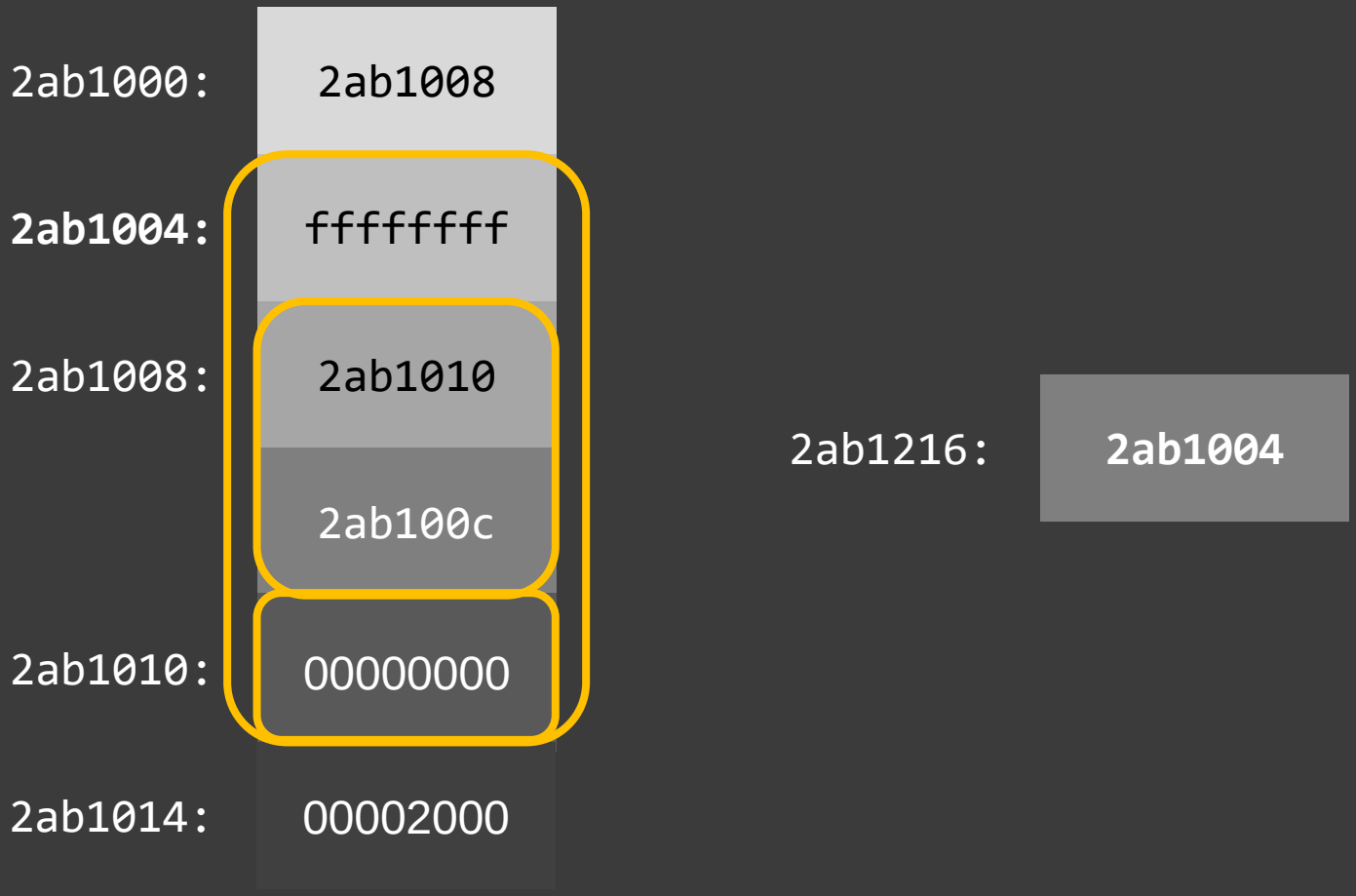




Addresses and Structures

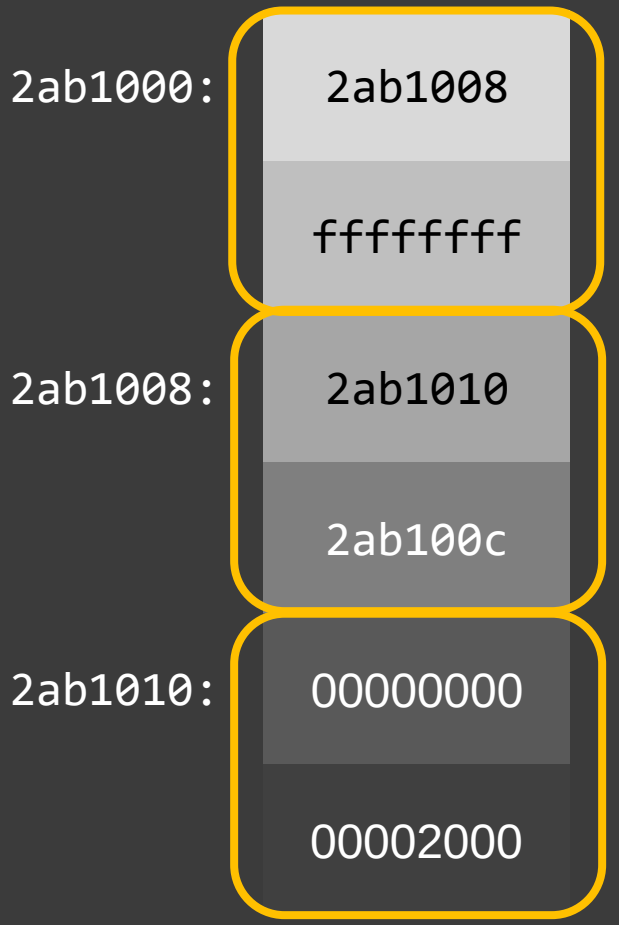


Pointers to Structures

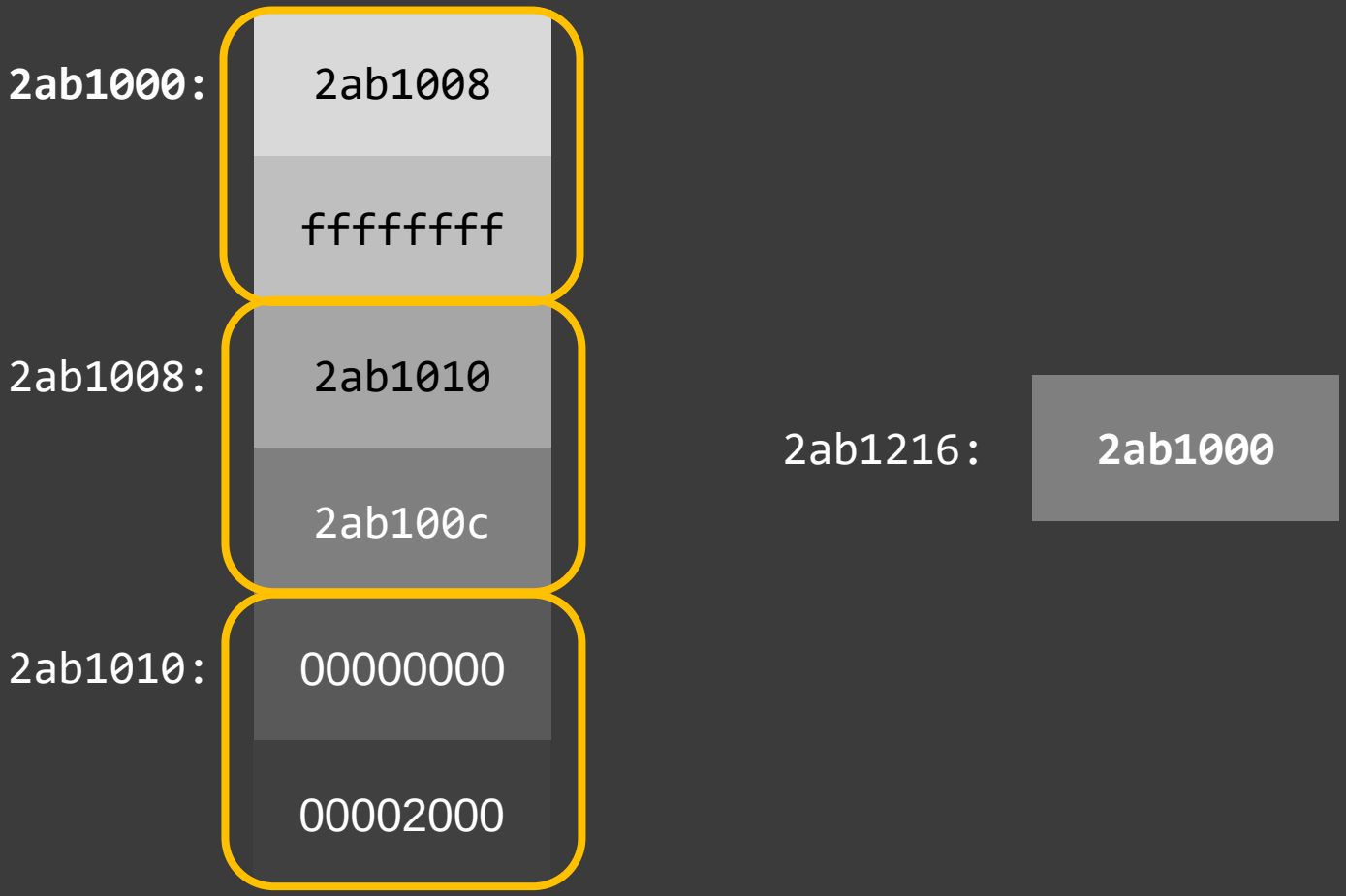


Arrays

2ab1000:	2ab1008
2ab1004:	ffffffff
2ab1008:	2ab1010
2ab100c:	2ab100c
2ab1010:	00000000
2ab1014:	00002000



Arrays and Pointers to Arrays



Strings and Pointers to Strings