

CONTENTS

Acknowledgments

page ix

1

Introduction

page 1

2

From Source File to Executable File

page 7

Transformation of a source file to a load (executable) module. Why we can and do discuss source programs and their behavior as if they were executing somewhere in memory in their source form. Concepts of static memory allocation, dynamic memory allocation, program address space, and program system stack.

3

Variables and Objects; Pointers and Addresses

page 21

Variables as “data containers” with names. Values as data – simple (in-nate or elementary) data, structures, and objects. Referencing variables

CONTENTS

through pointers. Unnamed “data containers” and their referencing through pointers. The dual role of pointers as address holders and binary code “interpreters”. Various interpretations of the contents of a piece of memory. Pointer arithmetic. Why C/C++ cannot be interpreted in a platform-free manner like Java can. Why C/C++ cannot have a garbage collector.

4

Dynamic Allocation and Deallocation of Memory

page 45

Fundamentals of dynamic allocation and deallocation of memory: free store (system heap); per-process memory manager; C memory allocators malloc(), calloc(), and realloc(); and C deallocator free(). How to handle memory allocation/deallocation errors.

5

Functions and Function Calls

page 59

System stack, activation frame, activation frame as the storage for local auto objects and for function arguments. Passing arguments by value as opposed to by reference. Calling sequence. Recursion and its relation to activation frames and the system stack. The price of recursion.

6

One-Dimensional Arrays and Strings

page 81

Static one-dimensional arrays and their representation as pointers. Array indexing as indirection. Why an array index range check cannot be performed in C/C++. The price of run-time array index range checking; the “compile-time checking” versus “run-time checking” philosophies. Passing static one-dimensional arrays as function arguments. Definition versus declaration of one-dimensional arrays. Dynamic one-dimensional arrays. Strings as static or dynamic one-dimensional char arrays terminated with NULL. How to add a custom-made run-time index range checker in C++.

7

Multi-Dimensional Arrays

page 97

Static multi-dimensional arrays and their representation. Row-major storage format and the access formula. Passing multi-dimensional arrays as function arguments. Dynamic multi-dimensional arrays.

CONTENTS

8

Classes and Objects

page 106

Basic ideas of object orientation; the concepts of classes and objects. Operators new, new[], delete, and delete[], and related issues. Constructors and destructors.

9

Linked Data Structures

page 132

Fundamentals, advantages, and disadvantages of linked data structures. Moving a linked data structure in memory, or to/from a disk, or transmitting it across a communication channel – techniques of compaction and serialization. Memory allocation from a specific arena.

10

Memory Leaks and Their Debugging

page 159

Classification of the causes of memory leaks. Tracing memory leaks in C programs using location reporting and allocation/deallocation information-gathering versions of the C allocators and deallocators. Tracing memory leaks in C++ programs: overloading the operators new and delete and the problems it causes. Techniques for location tracing. Counting objects in C++. Smart pointers as a remedy for memory leaks caused by the undetermined ownership problem.

11

Programs in Execution: Processes and Threads

page 187

Environment and environment variables, command-line arguments and command-line argument structure. A process and its main attributes – user space and process image. Spawning a new process (UNIX fork() system call) from the memory point of view. Principles of inter-process communication; System V shared memory segments and “shared memory leaks”. Threads and lightweight processes; advantages and disadvantages of threads over processes. The need to protect the “common” data in threads. Memory leaks caused by careless multithreading.

A

Hanoi Towers Puzzle

page 210

CONTENTS

B

Tracing Objects in C++

page 216

C

Tracing Objects and Memory in C++

page 227

D

**Thread-Safe and Process-Safe Reporting
and Logging Functions**

page 234

Glossary

page 239

Index

page 255