MACINTOSH DEBUGGING

Description:

Cut your development time and gain new insights into Macintosh debugging. “Macintosh Debugging” provides an introduction to source-level debugging and an in-depth look at machine-level debugging using MacsBug. The course explains how to locate and solve crashes that disable source-level debuggers. Additional emphasis will be given to “anti-bugging,” a technique for designing Macintosh applications whereby problems are detected before they can crash the machine.

Course Goals
During this class, you will:

• Review the use of source-level debuggers to locate errors in sample applications.
• Use MacsBug to locate errors in a sample application.
• Match object code (machine code) statements to the corresponding source-code statements.
• Identify procedure and function calls, global and local variables, procedure entries and exits, and A-Trap calls from MacsBug.
• Set breakpoints in testing an application.
• Add “anti-bugging” code that can catch errors before they crash the application.

Additional Benefits
• Practice debugging techniques in a supervised lab environment.
• Take home debugging reference materials and a source-code library of test and debugging tools.
• Examine additional debugging tools: TMON Professional, SADE, and others.
• Get tips on debugging stand-alone code resources and individual managers.
• Discuss how System 7’s virtual memory and 32-bit environment affect debugging and code development.
Learn techniques and tools you can use when debugging applications on the Power Macintosh. The class gives an overview of RISC and the Power Mac, and shows how to prevent errors and solve problems that are unique to this new system.

Facts:

Leader-led
4 days
$800

Target Audience:

C and Pascal programmers in the process of writing Macintosh software who desire help in debugging their products.

Prerequisites:

You should have at least 6 months of C or Pascal programming experience and have developed at least one Macintosh application that has windows and menus. You should understand the hexadecimal and binary number systems, Macintosh system error codes, pointers and handles, and heaps and stacks. A reading knowledge of 680X0 assembly language is highly recommended.

Course Outline:

I. Introduction
- A. Macintosh Hardware
- B RISC vs. CISC
- C. PowerPC Architecture
- D. Macintosh Error Codes
- E. Anti-Bugging
- F. Source-Level Debuggers (THINK C, Code Warrior, The Debugger, SADE, SourceBug, Macintosh Debugger for PowerPC)
- G. Low-Level Debuggers

II. Memory on the PowerPC Macintosh
- A. Low-memory globals
- B. The heap and the stack
- C. Stack frames and calling chains
- D. Understanding memory layout.
- E. Process Manager
- F Virtual Memory

III. PowerPC calling conventions and stack frames
- A. Emulator & Mixed Mode Manager
- B. Routine Descriptors & UPPs
- C. Registers & Stack Frames
- D. Cross-fragment calls and CFM calling conventions

IV. Using MacsBug
- A. Miscellaneous MacsBug Commands
- B. Locating where the application crashed
- C. User breaks

V. Understanding PowerPC Compiled Code
- A. MacsBug symbols
- B. Matching source and assembly
- C. Debugging C++

VI. PowerPC Lab
- A. Knowing your way around memory
- B. Understanding stack frames.
- C. Reading assembler
- D. Debugging practice.
- E. Understanding calling conventions.

VII. Macintosh Operating System
- A. Exceptions
- B. A-Traps, Patches, Packages, Glue

VIII. Debugging Tips & Tools
- A. Interrupts
- C. DCMDS
- D. ROM Debugger, Even Better Bus Error, TMON, The Debugger

IX. 68K Memory
- A. Memory layout between 68K and PPC
- B. non-VM systems

X. 68000 Assembly Language
- A. Code Segments & the Jump Table
- B. Registers
- C. Addressing Modes
- D.. Common assembly language instructions

XI. 68K Compiled code
- A. 68K instructions
- B. Matching source and instructions
- C . MacsBug Symbols

XII. 68K Stack frames
- A. 68K calling conventions
- B. OS Glue code
- C. Stack frame format
- D. Pascal and C formats.

XIII. A-Traps
- A. Trap Dispatch
- B. Patching traps.