Seminar Abstract

Advanced C

Course Overview:

This 2-3-day course teaches students advanced aspects of Standard C. The time is spent two thirds on lectures and one third on writing and debugging lab problems. To reinforce the theory, lab sessions are run immediately after the lectures to which they apply. The course is not hardware or operating system-specific.

Goals:

Provided students meet the prerequisites, at the end of the course, they should:

- Be knowledgeable about important but subtle issues regarding operator precedence and order of evaluation.
- Be comfortable with the type qualifiers invented by Standard C.
- Know how and when to use type synonyms.
- Be able to read and write arbitrarily complex declarations reliably.
- Know how and why to use pointers to functions.
- Know how to write an interrupt handler and how to register an exit handler.
- Be able to use the more advanced library functions and headers such as `setjmp` and `longjmp`, `assert`, and `stdarg`.

Who should attend:

Programmers and technical managers who need formal training in the more advanced aspects of C. Also experienced programmers not familiar with additions and changes made when C was standardized.

Prerequisites:

Successful completion of the 5-day "Introduction to C" course, or its equivalent, is assumed along with at least six months of serious C programming. Students are expected to be relatively fluent in the basic syntax and semantics of C, particularly with regard to data pointers, structures, and operators.

This course is definitely not for casual C programmers or those totally new to the language.

Materials:

Each student will receive the following materials:

- *Advanced C Language Seminar Handout.* This typeset manuscript has more than 300 $8\frac{1}{2} \times 11''$ pages. It contains a lot of useful material beyond that covered in the course.
- *Standard C Quick Reference Guide.* This 16-page guide was designed as a result of requests of students in earlier classes. It contains information such as keywords, operator precedence table, statement syntax, and a summary of the complete standard runtime library.

Detailed Topics:

The main topics covered are:

- Data pointer revision
- Operators, precedence, order of evaluation
- Sequence points and lvalues
- Comma operator
- The type qualifiers `const` and `volatile`
- `void` pointers
- Pointers to functions
- Pointers to arrays
- Type synonyms
- Function prototype issues
- Reading and writing arbitrary declarations with confidence
- Enumerated types
- Header design and preprocessor issues
- Details of the library headers `assert`, `signal`, `stdarg`, and `setjmp`.