Advanced C Programming

The C Programming Language was originally created to write the UNIX operating system. It quickly turned into a multi-purpose language used by all types of programmers for a wide variety of applications. C is a small language that can be learned quickly. It is highly structured and modular, supporting both small and large programs equally well.

This course fills the gap between an introductory course in C and more advanced application programming. Students write many programs, concentrating on data structures and file I/O.

Course Objectives:

- Describe the basic elements of C.
- Write C programs using all the major features of the language.
- Define and use C datatypes.
- Write variable declarations for programs.
- Apply the unique notations that C uses for assignments, incrementing, and decrementing.
- Control the flow of program execution.
- Write modular programs consisting of functions.
- Describe the purpose and functioning of a preprocessor.
- Define the relationship between arrays and pointers.
- Use structure variables for data storage and manipulation.

Audience: C programmers who need to advanced their coding skills.

Prerequisites: *C Programming*

Number of Days: 3 days

1. Course Introduction
- Course Objectives
- Overview
- Suggested References

2. The C Development Environment
- The cc(1) Command
- Include Files
- Libraries

3. Basic and Derived Data Types in C
- Simple C data types
- Integral data types
- Floating point types
- Derived data types
- Array data types - single and multi-dimensional
- Structure data types
- Simple pointer types
- Pointers to structures/multiple pointers
- Pointers to functions
- The const qualifier
- Bit operators
- Using typedef

4. Function: Calling, Passing, and Returning Values
- Anatomy of a function
- Parameter passing - pass by value
- Parameter passing - pass by reference

5. Standard I/O
- Standard I/O streams
- File access
- Formatted I/O
- String I/O

File positioning operations
Block I/O

6. **Low Level File I/O**
   Standard I/O vs. system I/O
   File access
   Direct I/O
   File Positioning
   Error Handling
7. **Memory Allocation with malloc and calloc**
   Dynamic memory allocation overview
   malloc(), calloc()
   realloc(), free()
   Structure Pointers
   Array of pointers to structures
8. **Memory Organization and the Scope of Variables**
   Command line arguments (argc, argv)
   The memory layout of a C Program
   The stack segment
   The heap segment
9. **Data Structures - Linked Lists**
   Array limitations
   Linked lists
   List operations - formation
   List operations - delete