

Symbian OS: Essentials and Systems Programming



Details

Level:	Intermediate
Duration:	5 days
Development language:	C++
Prerequisites:	None, reasonable understanding of C++ and OO/UML experience is beneficial. Familiarity with the essential basics of Carbide.c++ IDE is also beneficial.

Target audience

This course is for developers new to Symbian OS. The candidates gain a wide understanding and practical experience with the Symbian OS platform and learn the basic concepts to write software for Symbian OS phones using native C++.

The course consists of approximately 50% practical work and 50% lectures. The exercises include quizzes and C++ coding and use the TechView and Series 60 emulator to verify the results.

Description

This course is a shortened combination of the Symbian OS Essentials course and the Symbian OS Application Engine and Systems Programming. It covers a substantial part of the material from those two courses and it is pitched at an introductory level.

The course explores programming techniques required by both Symbian OS Systems Programmers (those developing and/or extending the operating system services) and developers working on application engines (services/functionality specific to an application).

The course trains programming techniques which are applicable to all Symbian OS phone types, all emulators, all UI systems and all versions of the OS.

These methodologies include:

- Symbian system overview
- E32 kernel architecture
- resource management
- descriptors and arrays
- file management (file server, streams & store, persistent objects)
- active objects and using client/server architecture

Within the system overview device families are introduced, the design of Symbian OS is motivated and the developer learns the philosophy behind the application framework and generic technology (GT) layers. While developing Symbian applications developers will heavily re-use components in the Symbian OS.

The E32 kernel architecture presents an overview of the multitasking capabilities in Symbian OS. We cover processes and threads, priorities, context switching, kernel service calls, stacks and heaps.

As memory and other resources are limited on a mobile phone it is crucial that developers handle the management of these resource carefully. Within the resource management section the developer gets familiar with Symbian concepts like cleanup stack, 2 phase construction, memory leak detection, etc to ensure your applications will run stable and free of memory leaks.

Descriptors and arrays are introduced as the Symbian way to deal with string handling (Unicode) and binary data. Besides saving on memory descriptors will help developer to identify the out-of-bounds problem easily.

Almost any application has to deal with data and files. In the file management section different options are presented to write and read data to files. The candidate gets an overview of the file server, streams & stores, object embedding and how to create persistent objects within Symbian OS.

Besides memory processor speed is also limited on mobile phones. The active object framework is presented in order to create single threaded applications. Active objects will save context switches and give better performance at lower clock rates and ultimately also save battery power. Active objects are also used as an introduction to client/server architecture. This course mainly explains how to use the existing servers in Symbian OS as a client.

During the course UML class diagrams are used to illustrate C++ coding concepts.

Objectives

- Develop simple programs for emulated environments and target devices.
- Use Symbian OS development tools competently
- Debug and test Symbian OS code (plus strategies)
- Understand the functionality of fundamental components of Symbian OS (kernel, system layer, application framework, design philosophy)
- Use the Symbian OS memory management model
- Use descriptors, arrays, streams, stores and the file system
- Implement and use active objects in a range of scenarios
- Recognize aspects of good system design
- Learn the differences between the three types of applications in Symbian: test or console applications and server (EXE), GUI application (EXE) and engines (DLL).
- Implement and use the standard Symbian services (e.g. libraries and servers)

Tools and platform

During the course we will use the Nokia Carbide.c++ OEM v2 IDE environment. Carbide.c++ can be downloaded from Forum Nokia free of charge.

Exercises and course materials are adapted for the Series 60 5th Edition SDK emulator.

A number of exercises will be run in the Symbian TechView UI. These exercises require features which are not present in the Series 60 environment or malfunction in the Series 60 environment.

Agenda

Day 1

Introduction to Symbian OS platform

- Key features
- Key platform components and their organisation
- Application architecture: Engine and Uis
- Overview of Symbian OS Development:
 - BAKs, DevKits, CustKits and SDKs: what they're used for.

Symbian OS Development Tools

- Kit installation, usage and documentation
- Basic Carbide.c++ commands
- Symbian OS build tools
- Unique identifiers
- Differences between target and emulated platforms

Application Development

- Building programs: GUI vs. console apps
- Using simple programs as templates
- Debugging and testing techniques
- Special Symbian OS coding conventions

Day 2

System structure

- Architecture and implications
- Kernel mode vs user mode operation
- Types of function calls (user, executive and server request)

Resource Management

- Overview of memory leaks
- Detecting memory leaks using the memory leak and alloc failure tools
- Two phase construction pattern
- Exception handling using TRAPD
- Using the cleanup stack for resource management

Descriptors

- Motivation for descriptors (compared with C strings)
- Descriptor classes and their key features

Arrays

- Dynamic array overview
- Use of segmented vs flat arrays
- Arrays for fixed/variable sized elements

Day 3

Active Objects

- Why AOs
- Life cycle of AOs
- CActive and CActiveScheduler classes
- Implementing simple AOs

Using Servers

- Client Server architecture overview
- Sessions and subsessions
- Associated cleanup issues

File Server and Stream Store

- Using file server sessions
- Basic direct stream store functionality
- Overview of different stream stores

Day 4

Application/Component Structure

- High level application/system design issues
- UI Engine and MVC application models

Building and testing

- Types of binary
- Writing DLLs
- Using RTest and test harnesses

Active objects

- Use cases
- Alternatives to active objects

Day 5

Implementing Servers

- When and how to write a server
- Design issues
- Implementing simple servers

Binary compatibility

- What is source and binary compatibility
- What causes binary compatibility to break
- How to tell if a change breaks binary compatibility
- How to avoid binary compatibility breaks in advance