

## Introduction to SMP

### Details

Level: Advanced  
Duration: 1 days  
Development language: C++



### Description

The migration to Symmetric Multiprocessing (SMP) is an exciting evolution of Symbian OS. This course introduces SMP, describes the changes necessary for existing code, and shows how to design multi-threaded code optimised for SMP hardware. This course consists of presentations with paper-based exercises to test the understanding of attendees of the content. There are no hands-on programming exercises.

### Objectives

This course will allow the participants to gain fluency in SMP concepts, and learn the skills required for developing high quality SMP compliant software.

Upon completing this course, participants will be able to:

- Appreciate the impact of SMP on Symbian OS and on Symbian software in general.
- Use their understanding of SMP hardware to think through SMP software issues.
- Use locks and other synchronization primitives appropriately.
- Make their code SMP Safe.
- Know when it is appropriate to develop software optimised for any number of processors and understand how to approach this task.

### Agenda

Introduction: Migration to SMP

- Symmetric Multi-Processing Basics
- What is meant by SMP Safe and Enhanced
- Motivation for SMP
- The new nanokernel

SMP Perspective

- Achieving scalable performance while conserving battery power
- Identifying components which will require re-engineering
- The need to run efficiently on SMP and non-SMP architectures
- The need to share memory impeccably

SMP Hardware and Architecture Essentials

- Overview of the many SMP designs
- Key aspects of the SMP processors on which Symbian OS will run

## SMP Software Essentials

- Inter-thread communication using a semaphore
- How to share memory safely using a mutex
- Other synchronisation options

## SMP Safe

- Identifying common uni-processor assumptions found in existing software
- How to pass data to a server asynchronously
- Avoiding reliance on thread priority ordering

## SMP Parallel Skills

- When to use multiple threads
- Parallelism: data versus functional, fine-grained versus coarse-grained

## Where to go from here

- Opportunities for further learning
- Finding more information