

## Porting Simkin to Symbian OS

Author: Simon Whiteside, <http://www.simkin.co.uk>  
Revision 1.0, July 2003

### Introduction

This article describes the major issues I encountered when porting Simkin to Symbian OS during April 2003. The aim is to share my experiences of the port, and provide help and advice to other developers making their software work on Symbian OS.

The article focuses mainly on the difficulties I experienced, as this is probably the most useful information for other developers. It is always painful to get to grips with a new platform, and Symbian OS is no exception. Having said that, it is always a very worthwhile experience. I found that Symbian OS has been very well thought-through, and provides a host of advanced features for the next generation of handheld devices.

After using Palm OS and Microsoft Windows CE, I certainly prefer working with Symbian OS.

Simkin (<http://www.simkin.co.uk>) is a script interpreter I have been developing since 1995. I first used it in an adventure game, then as the macro language for the well-known music notation program Sibelius (<http://www.sibelius.com>).

Simkin was originally developed in C++ for private use. After using it a few times myself, I decided to sell it as a library for others. I fairly soon realized that it would reach a broader audience if it was made Open Source, and since 2000 it has been available under the LGPL license. In 2000, I also developed a Java version of the interpreter and included support for XML.

Simkin is designed to be a lightweight language, which sits on top of an existing application. The application provides facilities that Simkin scripts can call into. Simkin has been used in a wide variety of applications, including games, skinnable user interfaces, test scripts and communications.

A useful aspect of Simkin, which creates a maintenance nightmare for me, is its cross-platform, cross-language interoperability. The library has been used on Windows, Windows CE, Macintosh and Unix platforms in C++, and within servlets, applications, applets and midlets in Java.

In April 2003 Symbian supported me in porting the C++ version of Simkin to Symbian OS.

### Reasons for the Port

There were several reasons for the porting exercise.

From the point of view of Simkin, it is useful to add another platform to those it supports. Some users already use Simkin within Windows CE applications, but were asking for the flexibility to move across to Symbian OS. I think that Simkin could be useful in the general goal of cross-platform convergence, and Symbian OS is an important platform within that. By "cross-platform convergence" I mean the ability for software to run across different devices, such as internet servers, desktops and handhelds, and for users to access the same functionality through these devices.

For Symbian, having Simkin on Symbian OS was useful in providing a powerful scripting capability to other apps developers. The port was an interesting experiment in showing that open source software from the wider world could be used on its platform. It was also hoping to garner information about the experiences of a developer in coming to grips with the platform (hence this article!)

From my personal point of view, I enjoy the challenge of learning about a new operating system. I have been a professional developer for over 15 years, and in that time have worked on many platforms, including Windows, Mac, Unix and OS/2. In the handheld area I have worked with the Sharp IQ, Windows CE and Palm OS.

## Porting goals

Given the relatively short timescale of the project (about a month) my porting goals were fairly limited. I wanted to port the library, while making as few code changes as possible. I also wanted to keep the programming interfaces intact – partly to keep the code and documentation clean, and partly to help existing users try out the new platform without changing too much of their code.

I also intended to produce one or two demonstration applications which would run on the phone. This helps users use the library. I also wanted to demonstrate Simkin to attendants of Symbian's Exposium03 conference.

## Getting to grips with Symbian OS

Prior to the project I had never developed in C++ for the Symbian OS. I had only read some theoretical papers, and parts of books. At the start of the project I did not have books on C++ development for Symbian OS, although I obtained these later. I started with a copy of Metrowerks CodeWarrior for Symbian OS, and the UIQ SDK for Symbian OS v7.0.

I found the SDK lacked conceptual information, so I read all the documents I could on the Symbian website, and other developer sites. This provided me with a basic grounding, and I was able to start doing basic porting work. Half-way through the project I attended a few days of "basic training" at Symbian, which helped fill in some of the gaps.

Something I found especially difficult was coming to grips with programming the user interface. I only needed basic functionality for my demonstrations, but spent many frustrating hours trying to make simple things work. Symbian provides the core of the operating system, out of which various versions are instantiated. The ones I targeted were "Series 60" and "UIQ". Neither of these came with useful conceptual information, and with rather sparse reference information.

In particular I found it very difficult to interpret the "Panic" codes that the SDKs displayed when error conditions were encountered. I spent many hours tweaking code in an attempt to try and work out why the toolkit was unhappy about what was happening.

The SDKs come with a small number of examples that don't fully exercise the APIs I was interested in. To help here, I downloaded large numbers of sample, and open source applications from the internet. This was a double-edged sword – I might see an API in action, but also learn a bad habit.

## Simkin porting issues

The overriding issue when porting Simkin to Symbian OS, was dealing with the particular limitations that are placed on how C++ can be used on the platform.

Symbian OS does not support C++ exceptions. Simkin for C++ was originally written to use exceptions. When porting Simkin to Windows CE, which also does not contain exceptions, I had to put in conditional code, which handled errors in a different way – so this was easy to cope with.

Symbian OS has its own notion of exception, called "leaving". This is essentially a traditional C `setbuf-longjmp` mechanism, where code control is transferred by changing the registers and the program counter instantly back to a certain point.

Unlike the C++ exception mechanism, these kind of exceptions do not unwind the stack. This means that the destructors of C++ objects are not called. Windows CE has a similar facility (called "structured exceptions"), which has similar consequences for C++ objects on the stack.

Unlike Windows CE, Symbian OS provides a mechanism for cleaning up objects left on the stack, called the "cleanup stack". This is a list of stack-allocated objects, which will be cleared in the event of an exception (or "leave") being thrown.

The Symbian OS cleanup stack is not supported by the C++ language directly. Programmers have to follow a carefully constructed list of protocols to make sure that their stack objects are cleaned up correctly. Program code has to explicitly place objects onto the cleanup stack, and remove them after use. Methods should be suffixed with "L" if they might cause a leave exception to occur – this indicates that extra care should be taken to put objects on the cleanup stack before the method is called.

An important part of the cleanup stack protocol is that the programmer should ensure that a leave exception is never thrown during the constructor of a C++ object – this seems to be a side effect of the lack of compiler support for the protocol.

For libraries, such as Simkin, written for "standard" C++, these protocols imply deep, and often subtle, incompatibilities with existing code. As well as implying that I had to carefully read and analyze every line within the library, to see if the protocols were being followed, it meant that I had to add special code for Symbian OS, and decide whether or not to suffix potentially leaving methods with "L".

In the end I opted for some compromises, which tried as far as possible to reach my goals of keeping a common code base, keeping the programming API intact and obeying the rules of the platform.

Specifically I did the following:

- i) where a constructor might leave, I `#ifdef`'ed it out for Symbian OS. I provided a non-leaving version, along with a leaving method which provided the same functionality. I then changed any calls within the library to use this "two-step" approach
- ii) I developed macros for putting objects onto the cleanup stack, which appear on all platforms, but compile to empty statements except on Symbian OS
- iii) I did not rename leaving methods, but clearly indicated in the method documentation (which is generated using Doxygen – <http://www.doxygen.org>) whether a method was leaving
- iv) I had to make all my classes derive from the Symbian OS class `CBase` for cleanup stack support to work correctly.

## Reference counted objects

The issues about exceptions had implications for two of the most important classes within the Simkin library: `skString` and `skRValue`. The first represents a C++ string, the second represents a variant value – which Simkin variables are bound to.

Both classes make use of reference counting. `skString` uses it to improve efficiency, by allowing strings to be passed by value without copying the underlying buffer. `skRValue` uses it to implement simple garbage collection. The Symbian OS cleanup stack has a simple approach to allocation and deallocation: an object is allocated, and put on the stack, then it is taken off the stack and deallocated. Reference counted objects have a different lifecycle – they are deallocated when the last reference to them is removed.

Fortunately Symbian OS had a means to deal with this – using the templated class `TCleanupItem`, it's possible to put objects on the cleanup stack and for a method to be called when the stack is cleaned up. For these classes I provided a method which decremented the reference count, finally deleting the object when the reference count gets to zero.

## Thoughts on exceptions

I think it is a shame that Symbian OS does not support standard C++ exceptions. The elaborate protocols significantly increase the skills required to program for the platform, and also the likelihood that errors will be introduced that are hard to track down.

It also makes it hard to port existing code to the platform, as all code must be carefully checked, and deep changes may be required.

Presumably the lack of support has something to do with performance, but I found that a lot of extra code was required to essentially provide the features that most C++ compilers include. My feeling was that it would have been more efficient in the long run for the C++ runtime to have provided the full C++ exception support – this would mean less special client code and a less complicated protocol for programmers to follow.

There are far fewer problems porting Java applications and libraries (such as Simkin) to Symbian OS, as Java has retained the support for exceptions.

## Symbian OS built-in classes

Simkin was developed before the widespread adoption of the C++ Standard Template Library. I'm also slightly wary of the compiler demands and incompatibilities introduced by using that library. As a result Simkin has its own set of strings (mentioned above) and container classes.

Symbian OS provides classes with analogous functionality. These have the benefit of following the leave protocol, and the Symbian OS naming protocol, which gives hints as to how to use classes and the cleanup stack. However they also have the drawback of not being cross-platform, and of being rather baroque and hard to understand (there are about ten different ways of representing a string, none of which provide an efficient, reference counted string!)

When porting Simkin to Symbian OS I chose to make only minimal use of the built-in classes. This took the form of some additional "convenience" functions and conversion operators, mainly in the `skString` class.

This has the drawback that the library contains more code than it needs to, but the advantage that very little of it had to be changed to run on the new platform. If Simkin becomes very popular on Symbian OS, it would be worth producing a particular version that used the built-in classes, and was therefore as small and efficient as possible. The Simkin classes make use of very standard C++ code, and functions from the C standard library. Fortunately most of these are provided by Symbian OS in its STDLIB package. There were some omissions from the package, and I had to provide my own versions of various functions, mainly connected with Unicode strings.

## XML support

Simkin is designed so that applications can store scripts in any format that is convenient. The library comes with support for storing scripts in TreeNode and XML files. TreeNode files are a simple text-based hierarchical format. These were easily ported across to Symbian OS.

Providing XML support was more difficult. At the time of the port, Symbian OS did not provide a single, general purpose XML parser, or set of XML document object model (DOM) classes. The UIQ SDK included some support, used by the SMIL package for multi-media messages (MMS). Unfortunately other applications could not use the XML classes, as crucial methods were not exported from the package DLLs. [These problems are fixed in GMXML component, which supercedes the SMIL composer and should be available shortly on DevNet. – Ed.]

The current version of Simkin for C++ supports various XML parsers and DOM libraries, including Apache Xerces, Microsoft XML and the popular open source Expat parser.

The Xerces framework is extremely complex, and makes massive use of C++ exceptions – and so seemed a bad choice for porting to Symbian OS. The Microsoft XML parser is available in binary form only, and only for Microsoft platforms.

Therefore I decided to port the Expat parser to Symbian OS, along with the DOM classes that Simkin includes which are used with this parser.

The Expat parser is written in plain C and was fairly easy to port. The main issues were connected with the initialized data problem (see below).

I'm hoping that once the Symbian strategy for XML support is finalized, that I will be able to revisit this, and integrate the new framework with Simkin.

## Compiler issues

I had a number of problems related to the different compilers used when developing for Symbian OS. I ended up using Microsoft Visual C++ and Metrowerks CodeWarrior when developing for the PC-based emulators, and the gcc-based compilers when building for actual phones.

These compilers had a couple of incompatibilities and particularities that made life difficult.

Firstly – they often had different interpretations of how to call conversion operators. The gcc-based compilers seem to prefer converting my `skString` objects to the `bool` type, rather than to something more suitable! These kind of errors are a real headache because they often only surface at run time on the device itself.

Secondly – the linkers for the device put the application data segment into read-only memory. This means that code cannot contain non-constant global variables. It proved tricky to construct global constructs, such as arrays or strings, which preserved their constant-ness. This error was shown as an initialized data error during the link. I had to search through the linker's `.MAP` file, and the source code to try and resolve the problem. This particularly affected the Expat parser library, as it makes wide use of static arrays.

## SDKs and development tools

I decided to produce the demonstration applications for two popular Symbian OS versions: Series 60 and UIQ. At the time I had to install the two SDKs on different computers. I have been told later by Symbian that it is possible to install them on a single computer, and that instructions for this are included in the SDK for Symbian OS v7.0.

It also proved difficult to produce conditional compilations from the same code base – as there weren't standard preprocessor flags indicating which user interface was being compiled against. I wanted to have one copy of the source code for the demonstrations, which could be compiled for either Series 60 or UIQ.

I've mentioned above that learning how to program for the two user interfaces was made difficult by poor documentation – I spent more time on this aspect of the project than on any other!

I used three of the main development tool environments for building Symbian OS C++ applications, finding strengths and weaknesses with each one. I prefer to use command line tools, and my trusty copy of the XEmacs editor. Symbian provides a good set of command line tools. Unfortunately these lack an easy way to do source-level debugging.

Another environment, which is being phased out, is Microsoft Visual Studio. This provides a debugger, and a quick compiler, but can't build applications for the device itself.

Symbian is moving towards using Metrowerks CodeWarrior – which can use its own C++ compiler for emulated builds, and the gcc-based compilers for the phone itself. Metrowerks provides a debugger, both for the emulator and the device. I didn't have the full version, so I wasn't able to test the device debugging.

Borland also provides C++ Builder, which targets Series 60. I haven't used this IDE for this project.

It would have been very helpful to have the SDK documentation provided in a more integrated way with the development environment – perhaps as hyperlinked Windows help files. Windows development is made very efficient in Microsoft Visual Studio, by being able to click on an API function and popup context help for the function.

## Symbian – the company

Symbian supported me in the porting of Simkin to Symbian OS. I found the individuals within the company to be very supportive, and was very happy to be invited to Symbian's Expositum03 to talk about Simkin.

I think they are having inevitable difficulties supporting developers using Symbian OS, which is compounded by the fact that several versions of the platform are produced and used by different phone manufacturers.

Hopefully, the availability of documentation and sample code will increase with time, and will become easier for developers from other platforms to get involved.

## The future

I hope that Simkin will prove popular on Symbian OS. If there is sufficient interest, and hopefully some funding support, I would like to produce a more efficient version of the library that uses the Symbian OS built-in classes, as well as the new XML parsing framework.

After the C++ port, I ported the Java version of Simkin to MIDP, which is supported by Symbian OS. A developer is already actively using this version to create a spreadsheet application!

I also market a source-level remote debugger for Simkin, and will be looking to make this work with the Symbian OS version of the library. This will involve me learning more about the TCP/IP capabilities of the operating system. Lastly I hope that the porting exercise will help me to get involved more with Symbian OS in my professional life as a freelance software developer. The operating system makes the mobile phone a very powerful device – when this is combined with always on network connectivity, it becomes a very exciting and compelling ingredient in developing computer systems.

## Bibliography

I was eventually able to obtain some Symbian C++ development books, which helped in the porting exercise. These were:

- Professional Symbian Programming (aka "The Red Book") – Wrox Press Ltd (2000) ISBN: 186100303X
- Symbian OS C++ for Mobile Phones – Wiley (2003) ISBN: 0470856114
- Programming for the Series 60 Platform and Symbian OS – Wiley (2003) ISBN: 0470849487

I made use of various websites as well, including:

- <http://www.symbian.com/developer/index.html> – source of various SDKs, a FAQ some articles and developer forums
- <http://www.ericsson.com/mobilityworld/sub/open/technologies/epoc/index.html> – provides information on UIQ and a good discussion board
- [http://www.forum.nokia.com/main/1,6566,1\\_32,00.html](http://www.forum.nokia.com/main/1,6566,1_32,00.html) – provides information on Series 60 and a discussion board

Want to be kept informed of new articles being made available on the Symbian Developer Network?  
[Subscribe to the Symbian Community Newsletter.](#)

The Symbian Community Newsletter brings you, every month, the latest news and resources for Symbian OS.