Writing Platform Independent Business Functions for OneWorld®

By John Oliver, Unity Enterprise Solutions

Introduction

One of the great advantages of the C language is its inherent platform independence. C compilers are available for almost every hardware and operating system imaginable. Why this article then? The reason is that while the C language is largely platform independent, the Operating System (OS) that hosts the software is not.

The OS provides a layer between the application software and the hardware platform. One of the main features provided by this layer is access to the disk file system and to the host's communications systems. These are two of the main paths through which OneWorld systems interoperate with –other systems.

It follows then, that the areas of interoperability and integration with remote systems throw up platform independence issues most often.

This article explores some issues that arise from this platform dependence and presents some approaches to overcoming them. Finally, a case study is presented for a Find Files function.

Why Bother with Platform Independence?

The answer is this: we generally would like to be able to write portable C code that can run on any platform we are likely to encounter. In the OneWorld environment, this typically means Win32 for Fat Clients, Citrix servers, and NT servers; and UNIX and OS400 for UNIX and AS400 Enterprise servers. In this article we will concentrate on the Win32 and generic UNIX platforms.

Typical Platform Independence Issues

As discussed, many system integration facilities provided by the OS are not platform independent. Some examples are:

File Systems

We are all familiar with the Win32 file system: Paths to files may be specified from a drive letter or from a URL, e.g., h:\mydirectory\myfile.csv, \\servername\sharename\hisdirectory\hisfile.csv. But UNIX systems use neither drive letters nor URLs. To make matters worse, the forward slash is used as a path separator, e.g., /home/username/hisdirectory/hisfile.csv.

Network Communications

On Win32 systems, reads and writes to network sockets are performed with the *send()* and *recv()* system calls. However, on UNIX systems the system calls *read()* and *write()* are used.

Synchronisation

Win32 systems provide amongst others, *WaitForSingleObject()* and *WaitForMultipleObjects()* to help provide synchronisation between threads. UNIX systems, depending on which thread library is used, typically use *pthread_join()* but have no simple equivalent to *WaitForMultipleObjects()*.

Approaches to Platform Independence

So how do we go about writing code to avoid the problem of Platform Dependence? Here are some approaches:

Copyright © 2003 by Klee Associates, Inc.

Avoid All Platform Dependent Routines

By avoiding the use of OS system calls that are not provided by all your target environments, the problem goes away all together. Unfortunately, as discussed, this is often impossible in System Integration projects.

Use an Abstraction Layer

An Abstraction Layer (AL) is a library of generic operations (typically available from 3rd party suppliers) that is available for a number of host platforms. By providing a common interface to all the functions, identical code can be deployed to all the supported platforms. The only difference is the binary AL library, which is platform specific. A different library is required for each supported platform.

Advantages

No special allowance is needed for Platform Dependant issues as long as the AL functions are always used.

Disadvantages

- The requirement to deploy the correct binary library for the AL adds to the complexity of a site's management.
- The cost of the AL
- The learning curve to become productive with the AL

Use Conditionally Compiled Code

The C compiler and language provide a means for the underlying platform to be identified at compile time via C macros. The C compiler has predefined macros such as __sun, __hpux, _MSC_VER and __ILEC400__, that indicate which platform is being compiled to. In OneWorld, the standard jdenv.h include file uses these macros to set one of its own macros such as JDENV_SUN, JDENV_HPUX, JDENV_PC, and JDENV_AS400. In our code, we can test these macros and conditionally compile an appropriate block of code. For example:

```
#if defined JDENV_PC
/* do some Win32 code here */
…
#elseif defined JDENV_UNIX
/* do some UNIX code here */
…
#elseif defined JDENV_AS400
/* do some AS400 code here */
…
#else
#error ERROR: This platform is not supported!
#endif
```

Advantages

No AL to purchase or learn.

Disadvantages

The code may become bloated with conditional constructs.

Case Study: a Find Files Function for Win32 & UNIX

Consider this scenario: a function is required to search a directory for files matching a pattern. This function is required for a UBE that may run on a UNIX Enterprise server, an NT Enterprise server, a Citrix server, or a Fat Client workstation. Iterative calls to this function return the file names one at a time.

Copyright © 2003 by Klee Associates, Inc.

JDEtips Knowledge Express

This Article Continues…

Subscribers, log in from our main search page to access the full article:

www.JDEtips.com/MyAccess.html

Not a Subscriber? Gain access to our full library of JDE topics:

www.JDEtips.com/JD-Edwards-Library

Visit **www.JDEtips.com** for information on the JDEtips University schedule, private training and consulting, and our Knowledge Express Document Library.

License Information: The use of JDE is granted to JDEtips, Inc. by permission from J.D. Edwards World Source Company. The information on this website and in our publications is the copyrighted work of JDEtips, Inc. and is owned by JDEtips, Inc.

NO WARRANTY: This documentation is delivered as is, and JDEtips, Inc. makes no warranty as to its accuracy or use. Any use of this documentation is at the risk of the user. Although we make every good faith effort to ensure accuracy, this document may include technical or other inaccuracies or typographical errors. JDEtips, Inc. reserves the right to make changes without prior notice.

Oracle and J.D. Edwards EnterpriseOne and World are trademarks or registered trademarks of Oracle Corporation. All other trademarks and product names are the property of their respective owners.

Copyright © by JDEtips, Inc.