Portionfest

Chinese Delight

page 2

Inside Dignus

64-Bit Development is Kickin'

pages3 - 4

Success Story

Professionals in Development

page 3

Diglets

Legacy nuggets

Every day now we hear about another IBM zSeries being installed and a whole host of mainframe software being decommissioned in favor of less expensive, cost effective replacement products. Large mainframe software companies are under tremendous heat to deliver lower cost solutions to their base of mainframe accounts. To that end, IBM has led the way with the Model z800.

IBM claims to have delivered more than 2000 units to 1500 sites. The list price of this product is $325,000; a far cry from $3.5 million for the traditional processor. So, paying $250K for a mainframe software product would seem absurd on the z800 relative to its older cousin. IBM once again is leading the field by providing a usage-based license fee that will set the stage for 2003. We'll stay tuned!

Winners Circle

Bob Shimizu & Dignus

Dignus, LLC is very pleased to announce that Bob Shimizu will be representing Dignus as an agent for its North American market.

Bob's professional background and knowledge of mainframe development will be greatly received in the market for Dignus leading-edge products.

Summer, 2002

Dave's Corner	3	Marketing	2
Winner's Circle	1	Compiler Lingo	4
Portionfest	2	Success Story	3

EASTERN BUFFET

Start with rice, add on sweet & sour chicken, throw in some fried wings. Next trip, pork skewers, bean sprouts, won ton soup, beef & mushrooms plus General Tso with noodles. Final plate, melon with Jello, almond cookies with custard. Oh, don't forget the fortune cookie if you can get it down. Chinese food from Eastern Buffet is a real stress buster and competes with Milton's for the Dignus tag team.

Marketing Dimensions

by Ron Pimblett

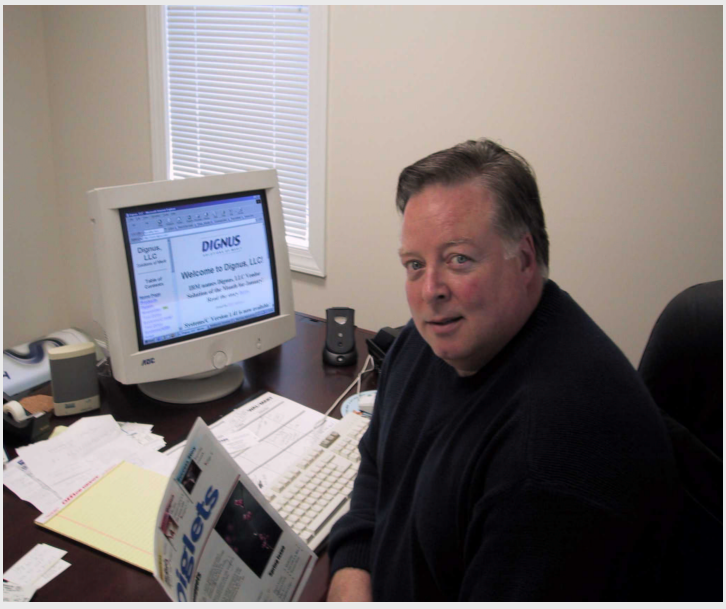
The Dignus booth will be well attended at the up-coming SHARE in San Francisco on August 19th. Once again, Dignus will give out the new IBM z/Series Reference Summaries and will be demonstrating the combined capability of robust Dignus Compilers, the Slickedit World Class Editor and the all encompassing Serena source management system at our joint Hospitality Suite on Tuesday evening, August 20th.

DIGNUS SOLUTIONS OF MERIT — slickedit. — serena

Systems Facilitating the art of programming — **Systems/C**

Systems ASM Facilitating the art of programming — **Systems/ASM**

Systems ++ Facilitating the art of programming — **Systems/C++**



Ubiquity

We are very pleased to announce our new reseller and partner in Australia. Ubiquity has a great reputation for support and represents, among others, Cole Software's XDC debugger and Serena, which Dignus is proud to interface to.

Dave's Corner

With the annoucement of Dignus' 64-bit compilers, we continue to demonstrate our committment to leading-edge mainframe development.

Dignus' 64-bit compilers allow for the most flexible transition to the 64-bit environment, while ensuring the best compatibility with other 64-bit hardware. This environment will ensure a seamless transition to the latest IBM and Linux mainframe operating systems.

Also, Dignus' 64-bit environment allows for continued use of 31-bit modules. So, the transition to the 64-bit environment need not be an "all or nothing" process.

We remain committed to your success!

Success Story

NEON Systems, Inc. has enhanced its development environment by adding Dignus 'state-of-the-art' compiler capability to its portfolio of development tools. This will allow Neon Systems to support and build applications more rapidly without compromise to the extreme scalability for which its software is universally recognized.

NEON Systems, Inc. (Nasdaq NEON) is the leading provider of Application Platform to Mainframe connectivity. NEON Systems technology is scalable, secure, robust, easy to use and provides a rapid return on investment. The result is enhanced cost-effectiveness and reduced burden on IT personnel. NEON also provides excellent customer support via the NEON Systems website, www.neonsys.com, 24x7 telephone lines, e-mail, and fax. For more information about NEON Systems and its solutions, call 800-505-NEON.

Compiler Lingo

by Greg Alexander

Exception Handling in C++

Suppose you're writing a procedure that is to initialize a pair of database connections, perform some transactions, then close them. The second connection fails. Your procedure (ideally) notices the failure, cleans up the first connection, then returns and signals an error condition to the procedure that called it. That procedure then needs to recover from its error condition -- if it also fails then the next procedure needs to handle it and so on. This chain of failures goes on until presumably some high-level control procedure takes a reasonable course of action, such as queueing the operation to be retried later.

Now enter C++. In C++ everything is considered to be an "object." What this means is that opening your database connection turns into creating a "connection object." So your procedure has created one "connection object," but in creating the second, a failure is encountered. That's a pretty exceptional event. So to indicate the failure, an exception is "thrown."

This means the C++ run-time library has to search through all of the procedures on the call stack looking for one that can "catch" the exception and deal with it appropriately.

The high-level controlling procedure has a "catch block" that is executed and it will be responsible for queueing the operation to be retried later. If it can't handle the situation, it could even "rethrow" the exception, meaning the whole process starts again from there. The tricky part comes when tracing the stack back to find a procedure to catch the exception. When it finishes searching a given stack frame (corresponding to one procedure), it has to clean up all of the objects in that procedure -- otherwise any memory, connections, or other resources it's allocated will remain allocated when the stack frame is discarded. To make this happen the compiler has to keep around a list of all the objects you may create so they can be destroyed. Not only this, it has to keep track of which ones have been created and which ones haven't. In the end, your C++ compiler records information about every object to handle all the "what if?" cases.

So all that cleanup and error checking you used to have to write by hand all happens "by magic" in C++, if everyone uses exceptions!