New Dignus Features to ease transition to z/TPF

Presented by
Dave Rivers, Principal Architect & Founder
Ron Pimblett, Marketing

DIGNUS

Dignus Product Milestones

- First Customer Shipment Systems/C in 1998
- First TPF customer deployment in 1999
- First Commercial Linux 390 compiler 2000
- Systems/ASM Spring 2001
- Systems/C++ 2002
- Local Linker for Cross Mode 2004
- Major architecture Release 1.80 2005
- IBM XP Link + CICS Preprocessor 2006
- DB2 Preprocessor 2007
- DBTE Basic Test Environment 2008
- TPF and zTPF compiler utilities 2009
- RDZ/Toolkit Plugins 2010

DIGNUS

Research & Development

- Member of IBM ‘Partners In Development’ Program
- Development Office in Raleigh North Carolina
- Dedication to the Development of Mainframe developer software for z/OS, TPF, z/TPF, z/VM…..
- Partnering with the TPF Lab

DIGNUS

Some of our Customers:

Neon Data Direct	Nationwide Ins.	Australia Securities
Serena Software	Sosa Cousins	Voltage Software
CONNX	IBM z/OS, z/VM	Anchor Software
EMC	Sun Storagetek	IBM TPF Lab
DIMIA	BMC Software	Referential Software
Software AG	Pitney Bowes	Softbase Systems
John Hancock	Montreal Transit	DesJardin Financial
GT Software	Barnard Software	Platform Solutions (IBM)
Data 21	Euroclear	Network Executives
CSI International	Marriott Group	Centrelink
Amadeus	Action Software	COPI
Australia Immigration	Manulife	Citizens Insurance
Neon Enterprise	SystemWare	HP

DIGNUS

TPF Customer Success

Amadeus

"What we particularly appreciate with Dignus is their compatibility with IBM's compiler. Mixing code compiled on IBM with Systems/C and Systems/C++ is transparent…just works," says Pierre Enault, Senior Systems Analyst ·

DIGNUS

IBM users and partnership

- IBM z/OS and IBM z/VM development
- IBM TPF Lab and z/TPF Evaluation
- IBM Tivoli Runtime implementation
- IBM Rational Developer joint Project
  German Beta Customer under evaluation

DIGNUS

IBM TPF Lab and z/TPF Evaluation

- IBM will certify support for the Dignus Compilers as an alternative to GNU C and C++
- Target Time Frames

  IBM Systems/C & Systems/C++ Testing Completed

  Certification announcement pending

  IBM z/TPF Beta program completed

  IBM GA prior to May 1, 2010

IBM Contact: Bob Dryfoos, TPF Lab Manager dryfoos@us.ibm.com

DIGNUS

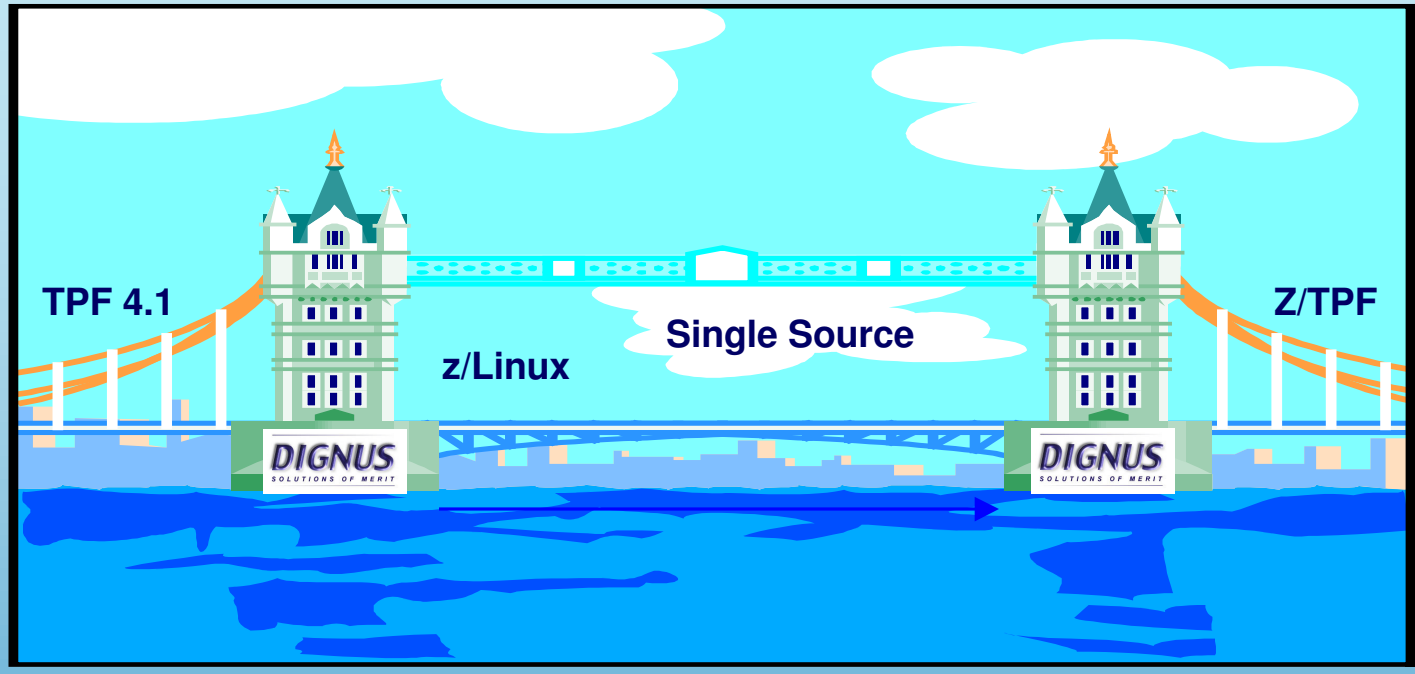
z/Linux

- Take advantage of cutting edge build environments
- Creates faster turnaround for developers
- Reduces z/OS cycles which saves $$$ Dollars
- Provides environment portability

*** Dignus cross mode allows the developer to run under Linux with all the same functionality as running under z/Linux

DIGNUS

Dignus recommended move to z/TPF



- Build on z/Linux for TPF 4.1 Some “single source” changes happen here
- Further “ single source” changes to accommodate z/TPF
- Build on z/Linux for z/TPF
- Retain 4.1 builds for fallback with no additional cost

DIGNUS

TPF & z/TPF Value Adds

- The ability to rapidly develop TPF programs for z/Linux or TPF 4.1 with professional compiler support while maintaining IBM compatibility.
- Cross Mode Compiling for faster turnaround
- Support for TPFGI and z/TPFGI (TPF Software)
- IBM Toolkit Interface
- Support for LCM Load Control Management (Sosa-Cousins)

DIGNUS

Latest Dignus Features

DCC & DCXX

- #pragma options (inline)
- #pragma inline (name)
- #pragma noinline (name)

DIGNUS

Latest Dignus Features

#pragma options (inline)

Attempts to inline functions instead of generating calls to those functions, for improved performance. When the INLINE compiler option is in effect, the compiler places the code for selected subprograms at the point of call; this is called inlining.

It eliminates the linkage overhead and exposes the entire inlined subprogram for optimization by the global optimizer. When the NOINLINE compiler option is in effect, the compiler generates calls to functions instead of inlining functions.

DIGNUS

Latest Dignus Features

#pragma inline (name)

“The z/OS #pragma inline directive specifies whether or not the function is to be inlined. The pragma can be anywhere in the source, but must be at file scope. #pragma inline has no effect if you have not specified the INLINE or the OPT compiler option. “ *

* IBM documentation source.

DIGNUS

Latest Dignus Features

#pragma noinline (name)

“The z/OS #pragma inline directive specifies whether or not the function is to be inlined. The pragma can be anywhere in the source, but must be at file scope. #pragma inline has no effect if you have not specified the INLINE or the OPT compiler option.” *

* IBM documentation source.

DIGNUS

Latest Dignus Features

DASM

- Support HLASM options via -options= '_ string_'
- Rename DASM to ASMA90 and it operates with the same command line options as ASMA90; integrates with maketpf

One difference:

IBM's ASMA90 produces an EBCDIC entry file, DASM produces ASCII; so only one small change needed to maketpf.

DIGNUS

DASM (Dignus Assembler)

IBM Support Statement

*** OEM assemblers available on the market that can run on the desktop**

"***IBM has used Dignus internally to the extent we are comfortable with it's compatibility. For problem reporting, IBM will treat customer code assembled by Dignus the same as HLASM***"

Ira Witkin, IBM Program Director

DIGNUS

New Dignus Features

- _Packed support
- #pragma map
- #pragma pack
- Rename header files with no source changes via $$HDRMAP
- _Decimal data types

DIGNUS

_Packed is fully supported, for example:

```
struct my_struct {
   char c;
   int i;
  } ;

  func()
  {
     struct my_struct unpacked;          /* normal alignment */
    _Packed struct my_struct packed;     /* packed alignemnt
    int i;

    i = sizeof(unpacked);
    i = sizeof(packed);
  }
```

Generates this code for z/TPF

```
# ***
# ***      i = sizeof(unpacked);
          LGHI  14,8          # 8
# ***
# ***      i = sizeof(packed);
          LGHI  14,5          # 5
```

DIGNUS

Note: Sizes of structures change based on the `_Packed` keyword.

_Packed example Structure Map from Dignus compiler listing:

```
    * * * * *   S T R U C T U R E   M A P S   * * * * *

================================================================================
| Aggregate map for: struct my_struct      Total size: 8 bytes                 |
|==============================================================================|
|      Offset       |      Length       | Member Name                          |
|    Bytes(Bits)    |    Bytes(Bits)    |                                      |
|===================|===================|======================================|
|       0           |       1           | c                                    |
|       1           |       3           | ***PADDING***                        |
|       4           |       4           | i                                    |
================================================================================

================================================================================
| Aggregate map for: _Packed struct my_struct        Total size: 5 bytes     |
|==============================================================================|
|      Offset       |      Length       | Member Name                          |
|    Bytes(Bits)    |    Bytes(Bits)    |                                      |
|===================|===================|======================================|
|       0           |       1           | c                                    |
|       1           |       4           | i                                    |
================================================================================
   * * * * *   E N D   O F   S T R U C T U R E   M A P S   * * * * *
```

DIGNUS

Note: Listing shows `_Packed` size and non-`_Packed`

#pragma map example:

Dignus implements #pragma map by generating the map'd-to name. For example:

```
#pragma map(a_very_long_name, "SHORT")
a_very_long_name()
{
  return 5;
}
```

Generates:

```
                    ...
#
                    .globl SHORT
                    .align 4
                    .type SHORT,@function
SHORT:
.Lbe24:
                    STMG  7,15,56(15)
```

Note: Function label name is "SHORT"

DIGNUS

#pragma map

IBM's solution to `#pragma map` is to introduce a duplicate label. This allows a reference, but does not actually change the original name. Programs can reference the new name, but the "original" name continues to exist causing potential problems:

IBM C source with `__asm__`

```
__asm__("SHORT:");
int
a_very_long_name()
{
  return 5;
}
```

GCC Generates:

```
#APP
        SHORT:
#NO_APP
.text
        .align  4
.globl a_very_long_name
        .type   a_very_long_name, @function
a_very_long_name:
.LFB2:
        stmg    %r11,%r15,88(%r15)      #,,
```

Note: duplicate symbols

DIGNUS

pragma pack

IBM has the following solution for changes to support #pragma pack:

Complete the following steps to convert #pragma pack statements:

Convert each #pragma pack(packed) statement to #pragma pack(1).

Convert each #pragma pack(twobyte) statement to #pragma pack(2).

Convert each #pragma pack(full) statement to #pragma pack(4).

For the GCC compiler, the #pragma pack( ) statement returns to natural alignment. However, for the z/OS compiler, the #pragma pack( ) statement is a 4-byte alignment and the #pragma pack (reset) statement returns alignment to the previous rule. The best way to handle these different meanings is to do the following:

Convert each #pragma pack( ) statement (that is, a statement that has no value specified between the parentheses) to #pragma pack(4).

Add a guard macro (#ifdef __370__) before each #pragma pack(reset) statement. Add #else, #pragma pack( ), and #endif statements after the guard macro.

Dignus supports the IBM C compiler #pragma pack semantics, no changes required.

DIGNUS

Decimal data types supported in z/TPF:

Example:

```
func()
{
    _Decimal(5,1) d,q,r;

     q = 1;
     r = 2;

     d = q+r;
}
```

Generates:

```
    ...

        BNZ   4064
# ******* End of Prologue
# *
# ***      _Decimal(5,1) d,q,r;
# ***
# ***       q = 1;
        LA    14,.LC0-.LT0(13)
        MVC   539(3,15),0(14) # q
# ***       r = 2;
        LA    14,.LC1-.LT0(13)
        MVC   542(3,15),0(14) # r
# ***
# ***       d = q+r;
        MVC   501(3,15),539(15)
        XC    500(1,15),500(15)
        AP    500(4,15),542(3,15)
        MVC   517(3,15),501(15)
        MVC   536(3,15),517(15) # d
# ***   }
        .align 2
.L0:

        …..

 .size func,.Lfe24-func
 # * **** End of Epilogue
        .align 4
.LC1:    .byte 0x00
        .byte 0x02
        .byte 0x0C
        .byte 0
.LC0:    .byte 0x00
        .byte 0x01
        .byte 0x0C
        .byte 0
#
```

DIGNUS

Note: Use of decimal instructions & data

$$HDRMAP

Maps a source #include name to another; useful for moving between a mainframe and z/Linux environment with no change to the source.

Example $$HDRMAP:

"MYPDS(MEM)" my_directory/mem.h

DIR cinc my_directory

`#include "MYPDS(MEM)"` would actually look for 'my_directory/mem.h' as if the souce had #include "my_directory/mem.h".

`#include "cinc/inc.h"` would look for "`my_directory/inc.h`".

Other Dignus compiler options that obviate changes:

-faddh: adds missing .h appendix to file names
-flowerh: Changes .H to .h
-fincstripdir: Removes any directory component
-fincstripsuf: Removes any suffix
-fincrepsuf: Replaces any suffix with .h

DIGNUS

Note: No changes to original source to accommodate different directory structure.

Dignus Partners for z/TPF

Sosa Cousins

LCM Load Control Management

- Coordinates development and promotion process
- Leverages extensive metadata to ensure quality
- Auditing facilities track source versions
- Designed specifically for TPF development – yet flexible and extensible for development for other systems (e.g., Websphere, z/OS, etc.)

DIGNUS

Dignus Partners for z/TPF

TPF Software

TPF/GI®

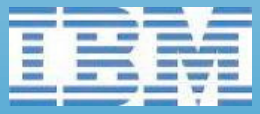
•Interactive Client/Server Testing for TPF
•Seamless GUI testing and debugging for C, C++, Assembler and SabreTalk
•The best of the PC and TPF Worlds

DIGNUS

Dignus Partners for z/TPF

TPF Toolkit

- The TPF Toolkit source scan tools that help you convert TPF 4.1 application code to single source and then maintain this single source until your migration to z/TPF is complete.
- Develop, deploy, and manage Web Services for z/TPF in a top-down approach.



DIGNUS

*registered trademark of IBM

Dignus Summary

- Integration with TPF/GI, LCM & IBM Toolkit
- IBM plug Compatibility
- TPF 4.1 & z/Linux support
- Cross Mode for faster compiles
- z/Linux migration aids to assist in bldg TPF on z/Linux
- Exceptional Customer Support

DIGNUS

NEXT STEPS

Questions??

Request:

- Free Trial ** Try using our trial to see how Dignus can speed your implementation to z/TPF.

DIGNUS

Thank you for coming !

DIGNUS