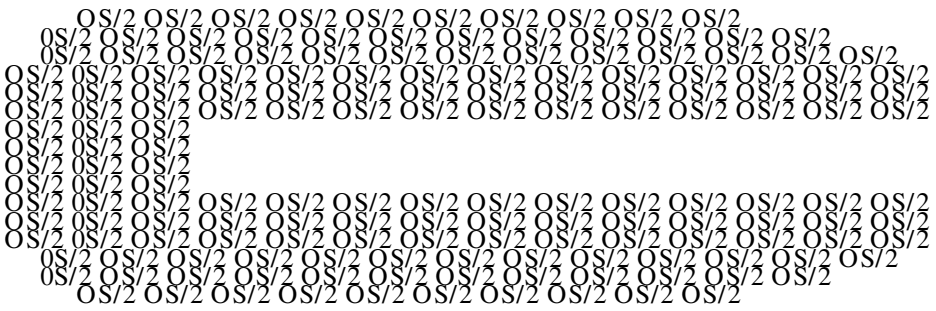

IBM OS/2 Compiler News

November 1992 No. 3

In this issue

C Set/2 .. PLUS?....

This issue we're delighted to discuss the beta program for IBM's C++ for OS/2.[1] You'll read what we're making available, and how to join in the test. Also in this issue:

plus of course our usual quota of Q&As, page 19

Where did it go?

We've received quite a lot of mail concerning the 'missing' July issue. Did it go astray? No: we didn't have one, although with issues dated January and April, we can't blame you for thinking there would be one. We try to publish on a roughly quarterly basis, but things don't always go the way we want them to, and we didn't publish a summer edition. We're sorry we worried you, but we're gratified to hear you missed us! Starting with this issue, to make sure you know where you are, we are numbering the newsletters now as well as dating them.

---

[1] ™ OS/2 and products marked ™ are trademarks or registered trademarks of IBM Corporation

[2] ™ Trademark or registered trademark of CompuServe Inc

Calling all C++ fans ...

What do computer companies and the Greek philosophers have in common? They both make extensive use of the Greek letter beta. In our case, it's our long-awaited OS/2 C++ beta. We're delighted to confirm that the much circulated rumour of an IBM C++ for OS/2 is true. The product's beta version has been announced and it's all ours, written in-house by the folks who gave you C Set/2 and WorkFrame/2.

Two questions come immediately to mind: what's in it? and how do I get it? And if you didn't know about the beta, you're probably asking where the announcement was made. For the content of the beta, see the following article; everyone else stay here.

The C++ beta program was announced by John Soyring, IBM's OS/2 Director of Marketing. At the SD92 (Software Development '92) show in September, Soyring stated that IBM was enhancing what we call the "DAP" - the Developers Assistance Program - to reach software developers anywhere in the world who are interested in developing OS/2 applications. Joining the DAP is free of charge,[3] and amongst many other benefits, you get our beta and a whole bunch of other beta materials - all on one beta CD-ROM.

CD-ROM? Yes, you read that right: our beta code and documentation come on an OS/2 CD-ROM disk at no cost (some countries may charge a nominal shipping and handling charge; this is $15 in the US). Join the DAP, and the CD-ROM is yours. Availability is mid-November.

Developers in the US can apply for DAP membership by calling (407) 982-6408; you'll receive a sign-up form via fax. Outside the US, you should contact your country DAP co-ordinator or local IBM office.

You can also apply electronically by typing "GO OS2DAP" on CompuServe and completing the online sign-up form.

[3] There is no charge to join the DAP but some program services and product require a fee.

Beta code: what you'll get

We cannot stress too highly that this is beta code, not the finished product, but even so, we know you'll like what you see. The product is based upon the very successful IBM WorkSet/2™ suite of products (C Set/2™, WorkFrame/2™, and Programmer's ToolKit™), and has the following features:

- ANSI conformant

  X3J11 for C, and the draft X3J16 for C++ with Templates and Exception Handling.

- AT&T

  AT&T language level 3.0 supported for compatibility.

- CUA™ 91

  CUA91 means usability through standardization of the Graphical User Interface. The human factors principles embodied in this standard, should enable users to benefit from improved ease of use, faster learning, increased productivity, reduced error rate, and overall increased satisfaction.

- National Language Support and Double Byte Character Set enabled..

The product comprises:

- C/C++ compiler

  An ANSI/ISO/SAA conformant highly optimized compiler, with extensions for OS/2 and migration support.

- Interactive source level debugger

  Enhanced to handle specific language constructs, e.g. classes

- Class Libraries
  - Basic libraries

    I/O Stream, Tasking and Complex from AT&T, ported to OS/2
  - Class library: Collection Classes

    The Collection Classes offer a full range of basic abstract data types like sequences, sets, maps etc.. To provide optimal performance they are based on many implementations ranging from simple arrays or linked list to trees or hash-table. Consistent interfaces allow to change the implementation variants without recoding.

- User Interface (PM)

  Provides an object-oriented encapsulation of PM, giving a programming model more suited to C++ language. This helps reduce the complexities of PM user interface development by offering a higher level of abstraction, through libraries of classes that you can use and easily extend.

- Execution Trace Analyzer

  We offer an OS/2 based execution trace analyzer that helps you to improve and understand the behavior of C and C++ programs. It traces the execution of a program and then displays the analysis using various diagrams. It can be used to help improve a program's performance, to examine what led to certain types of faults, and, in general, to understand what actually happened when the program ran.

Browser function

This is the package as it stands today, but of course, it's only beta level code. If you look at the RISC/6000™ C++ product, you will see a browser, and you'll note we never mentioned it above. As stated at various presentations and demos, it's our intention to offer such browser functions with OS/2 C++.

Hello, IBM .... hello, World?

Is there anybody out there, you may ask. Well, there certainly is on CompuServe. Ian Ameline, one of the designers and developers of the optimising code generator in C Set/2, is our man online. He monitors Section 4 of the OS2DF1 forum on this service, the home of C Set/2. Ian is logged at least twice a day, and welcomes the opportunity to talk to people. He handles bug reports, how-to's, and then, if something specialised comes up, behind Ian is the entire WorkSet/2 product group. Queries are turned around as quickly as possible, and fixes are made available in Library Section 4. Check us out.

Reading up on OS/2

Want to curl up with a good book? Take a look at some of these if your choice of reading matter runs to the technical: they've all been published within the 12-18 months. It's not an exhaustive list but it's a good selection for all types of reader. Stay tuned for more titles in future issues.

The information has been provided by the authors and their publishers. IBM author information, IBM order numbers and internal Mechanicsburg prices have been added where appropriate.

THE DESIGN OF OS/2 by H. M. Deitel and M. S. Kogan

400 pp., hardcover. Published by Addison-Wesley, November 1991. ISBN 0-201-54889-5 - $40.95 US. IBM order number S325-4005 - $40.95.

The primary goal of The Design Of OS/2 is to provide insights into the design decisions and philosophy of the OS/2 operating system. It discusses the motivation, architecture, and realization of OS/2 in the personal computing marketplace. The design of the major components of OS/2 are described in terms of their API architecture, internal data structures, and algorithms. Each area focuses on bridging operating systems theory to the realization of the design and implementation of OS/2. Where it is significant, an objective comparison of the technical aspects of OS/2 and other operating environments is provided. A key thrust is to describe the evolution of personal computer operating systems from DOS through 16-bit OS/2 and 32-bit OS/2.

CLIENT-SERVER PROGRAMMING WITH OS/2 2.0 by Robert Orfali and Dan Harkey

Both authors are with IBM, San Jose, CA. Published by Van Nostrand Reinhold. First Edition, June 1991, Second Edition (V 2.0) due in July 1992. 1,026 pages. ISBN 442-00826--0 - $39.95 US. IBM order number G325-0650 - $34.00.

Guide to Client-Server programming and OS/2 Extended Edition / Extended Services. Includes: database servers vs. transaction servers; APPC, NetBIOS, Named Pipes, LAN Server; remote vs. local SQL vs. stored procedures; OLTP, database and communication benchmarks; GUI tools and rapid prototyping; plus 300 pages of C programming examples.

Second Edition adds: how the client/server model is evolving; new OS/2 2.0 and Extended Services features; TCP/IP, Novell and CPI-C; the OS/2 2.0 Workframe programming environment; 32-bit C Set/2 compiler; case study: CUA '91 object-based front-end with a transaction server back-end.

OS/2 PRESENTATION MANAGER GPI: A Programming Guide to Text, Graphics, and Printing by Graham C. E. Winn.

Graham is at IBM Hursley, UK. Published by Van Nostrand Reinhold, June 1991. 318 pages. ISBN 442-00468-0 - $39.95 US. IBM order number G362-0005 - $33.67.

This book helps you create superior OS/2 graphics using the powerful graphic programming interface by clarifying a complex package that has not been well documented or understood. Over 50 programming examples are given that help answer questions on GPI and development functions. All aspects of data transfer, deletion, dimension functions, page units, scaling of MetaFiles and form selection are included.

WRITING OS/2 2.0 DEVICE DRIVERS IN C by Steven J. Mastrianni.

Published by Van Nostrand Reinhold, April 1992. 250 pages. ISBN 0-442-01141-5 - 36.95 US. IBM order number G362-0006.

This book provides the skills needed to write device drivers: it shows how operating system components interact and how device drivers fit in; compares 16-bit drivers to 32-bit versions; DOS compatibility; mouse drivers, storage devices, Micro Channel, tools, debugging, tuning, and performance; virtual device drivers; memory mapped adapters and IOPL; direct memory access; Extended Device Driver Interface; device helper reference.

USING OS/2 2.0 by Barry Nance with Greg Chicares.

Published by Que Corporation, April 1992. 470 pages. ISBN 0-88022-863-6 - $24.95 US. IBM order number G362-0007.

A guide to understanding and learning to use OS/2 2.0 and a reference for first-time computer users, as well as DOS and Windows pros. Installation and configuration; command reference, Workplace Shell; built-in help; Enhanced Editor; running DOS, Windows, and OS/2 programs; using the tutorial and built-in help; troubleshooting; batch files; printing.

INSIDE OS/2 RELEASE 2.0 by Staff of New Riders Publishing.

May 1992. 850 pages. ISBN 1-56205-045-1; $34.95 U. S. IBM order number G362-0016. Comprehensive guide to OS/2 2.0 for the advanced DOS or Windows user making the move to OS/2. Describes in detail the advantages of OS/2 over competing operating systems from the end-user perspective. Describes and explains the OS/2 graphical user interface, customizing the desktop, and built in

applications. Extensive descriptions of requirements to run DOS or Windows applications under OS/2 and surveys significant OS/2 applications. Additional chapters discuss networking under OS/2, optimization, and batch programming with REXX.

OS/2 APPLICATION DEVELOPMENT TOOLS by Brian Proffit

220 pages. Published by Premier Graphics, May, 1992. $22.95 US. IBM order number G362-0004 - $19.38.

A comprehensive review of development tools for OS/2. Describes the evolving history of the software development process, OS/2's unique benefits to the application developer, and valuable tips and techniques on the tool selection process.

NOW THAT I HAVE OS/2 2.0 ON MY COMPUTER -- WHAT DO I DO NEXT? by Steven Levenson and Eli Hertz, Hertz Computer Corp.

Published by Van Nostrand Reinhold, June 1992. 300 pages. ISBN 0-442-01227-6 - $24.95 US. IBM order number G362-0008 - $21.03.

The OS/2 2.0 Workplace Shell: interacting with the operating system and applications; customizing the environment; common functions; comparisons to DOS and Windows; FAT and HPFS file systems; object manipulation and customization; guided tour; applets; printing and fonts; editing the CONFIG.SYS file; built-in and third party diagnostic and troubleshooting utilities; tips and techniques.

The OS/2 2.0 USER'S GUIDE FOR THE WORKPLACE SHELL by Maria Tyne.

Published by Computer Information Associates, June 1992. 300 pages. ISBN 1-881435-47-4 - $24.95 US. IBM order number G362-0014.

Learning and understanding the Workplace Shell; for novices and experts: visual examples (over 100 screens); step by step procedures; a quick reference. A quick and easy introduction to the OS/2 2.0 environment: windows, objects, applications, folders, and information help.

INTEGRATING APPLICATIONS WITH OS/2 2.0 BY WILLIAM H. ZACK.

Published by Van Nostrand Reinhold, Summer 1992. 300 pages. ISBN 0-442-01234-9 - $34.95 US. IBM order number G362-0009.

Implementing and using applications in OS/2 2.0; migrating DOS and Windows applications to OS/2 2.0; Workplace Shell graphic user interface; "applets"; internal OS/2 management tools; installation and customization; hardware requirements; memory management; multitasking; program management; file systems and file management.

THE COBOL PRESENTATION MANAGER PROGRAMMING GUIDE by David M. Dill.

Published by Van Nostrand Reinhold, June 1992. 280 pp. ISBN 0-442-01293-4 - $38.95 U.S.; IBM order number G362-0010, $32.82.

Introduces OS/2 programming through easy-to-understand COBOL examples, with skeleton code. Especially useful for COBOL programmers wishing to migrate applications from a mainframe to OS/2.

LEARNING TO PROGRAM OS/2 2.0 PRESENTATION MANAGER BY EXAMPLE by Stephen A. Knight

Stephen is with IBM Rochester, MN. Published by Van Nostrand Reinhold, July 1992. 200 pages; $39.95 U.S. IBM order number G362-0011. For application programmers new to the PM environment. The book develops an actual 32-bit OS/2 2.0 application program, explaining each step of writing and compiling the code. This example illustrates many of the Presentation Manager features and application program interface (API) calls.

COMPREHENSIVE DATABASE PERFORMANCE FOR OS/2 2.0's EXTENDED SERVICES by Bruce Tate, Tim Malkemus, and Terry Gray.

The three authors are with IBM, Austin, TX. Published by Van Nostrand Reinhold, September 1992. IBM order number G362-0012.

A complete guide to database performance, from database design to optimizing queries to benchmarking a completed application. Sample benchmarking code, a symptom / response troubleshooting matrix and a list of practical tips and techniques.

OS/2 UPDATE by Dick Conklin

Dick works for IBM, Boca Raton, FL. Published by Van Nostrand Reinhold, Fall 1992. IBM order number G362-0015.

A collection of popular articles from IBM's Personal Systems Developer magazine, 1990 - 1992. Picks up where OS/2 Notebook (G362-0003) left off.

C PROGRAMMING IN THE OS/2 2.0 ENVIRONMENT by V. Mitra Gopaul.

Published by Van Nostrand Reinhold, November 1992. IBM order number G362-0013.

Abstract not available.

Canada, eh?

Many of the newsletter reply forms come back with questions like: who the heck are you guys? who writes this newsletter? are you really IBM? Well, here's the scoop: yes, we're IBM. We're part of the Language Technology Centre of IBM Canada Laboratory, and are more familiarly known as the C++ Compiler Folks. We're in Toronto, home of the World Series Baseball champions, The Toronto Blue Jays.

The LTC has over 300 people working on the development of compilers and compiler tools for all IBM platforms, from our own beloved C Set/2 on OS/2, through C++ and Fortran on the RISC System/6000, to C on System/390, and, of course, the AS/400™ where we have a complete suite of languages, including RPG and COBOL.

Our mission? The creation of state of the art development tools using the best of IBM's R&D worldwide. Check out our C++ beta code: the Execution Trace Analyzer, has a graphics base that came from our TJ Watson Research Lab in Hawthorne, New York; our container classes are developed in our software lab in Boeblingen, Germany; our Lexington software lab in the state of Kentucky provides the debugger and the Execution Trace Analyzer; browser development has been handled by the research lab in Tokyo. And for C++, the cement of these many bricks is Toronto, Canada.

As for the OS/2 C/C++ products themselves, one thing you may not realize about the WorkSet/2 products is that they were created for developers by developers, and we're our own customers - we use them every day, day in, day out. Hundreds of products being devel-

oped in IBM all across the world, those for internal use only and those intended for sale, depend upon the WorkSet/2 product suite. If something doesn't work for you, it doesn't work for us, either. Thus we have a very selfish interest in making our product the best we can, and fixing anything that doesn't work!

And who writes the newsletter? We'll never tell .....

Ahead, warp factor ....

Are you going as fast as you'd like with your C Set/2 programming? Here we review some ways to improve your program's performance. We do so with the caveat that optimizing for speed may result in an increase in the size of your executable module. Read the article immediately following this one, which covers decreasing .EXE size, and decide what's best for you.

Choosing Compiler Options

The following list names the compiler options that can improve performance. It also describes what each option does to cause the improvement.

Option Effect

/Gf+ Generates code for fast floating-point operations.

/O+ Turns on optimization. The compiler always optimizes for speed.

If your program has only one thread, use the /Gs+ option to disable stack probes. Because the stack of the first thread is always fully committed, stack probes are not necessary in single-thread programs.

If you link your .EXE files in a separate link step, specify the /BASE:65536 linker option to tell the linker your .EXE will be loaded at 64K. The linker can then resolve a number of references that would otherwise be left for the loader to resolve at run time. When you use icc to link your program, it specifies this option for you by default.

Note: Do not use the /BASE:65536 for DLLs.

The following options do not improve the performance of your code so much as they prevent the generation of objects or information that can degrade performance. Note that these are set by default:

Option Effect

/Gh- Does not generate profiler hooks.

/Gr- Generates code to run at ring 3. If you use /Gr+ , the code generated runs at ring 0, and the performance may suffer.

/Gw- Does not generate an FWAIT instruction after each floating-point load instruction.

/Ti- Does not generate debug information.

/Ts- Does not generate code to allow the debugger to maintain the call stack. (in CSD 22).[4]

Choosing Libraries

Your choice of runtime libraries can affect the performance of your code:

- Use the subsystem library whenever possible. Because there is no runtime environment for this library, its load and initialization times are faster than the other libraries.
- If your application has multiple executable modules and DLLs, create and use a common version of a runtime library DLL.

Allocating and Managing Memory

The following list describes ways to improve performance through better memory allocation and management:

- Use _alloca to allocate dynamic storage that is specific to a function. Because _alloca allocates from the stack instead of the heap, the storage is automatically freed when the function ends.
- You can use either malloc or DosAllocMem to allocate storage. Use malloc if you plan to reallocate the storage with realloc. Otherwise, use DosAllocMem for better performance.
- For maximum performance ensure all objects that require doubleword alignment are stored into dynamically allocated memory such that they **are** doubleword-aligned. Note: malloc, calloc, and realloc provide storage that is doubleword-aligned.

Using Strings and String Manipulation Functions

The handling of string operations can also impact the performance of your program:

- Use #pragma strings (readonly) to make your strings read-only. If you use the intrinsic string functions, the compiler can better optimize them if it knows that any string literals they are operating on will not be changed.

---

[4] CSDs are available through CompuServe, or through your local IBM Level 1 support. The number does not mean there have been 22 CSDs; it's just a build number. Also remember that CSDs are cumulative: you need only the last one.

- When you store strings into storage allocated by malloc, align the start of the string on a doubleword boundary. This alignment allows maximal performance of the string intrinsic functions. The compiler performs this alignment for all strings it allocates.
- Keep track of the length of your strings. If you know the length of your string, you can use memcpy instead of strcpy The memcpy function is faster because it does not have to search for the end of the string.
- Avoid using strtok. Because this function is very general, you can probably write a function more specific to your application and gain better performance.

Performing Input and Output

There are a number of ways to improve your program's performance of input and output:

- Use binary streams instead of text streams. In binary streams, data is not changed on input or output.
- Use the low-level I/O functions, such as _open and _close. These functions are faster and more specific to the application than the stream I/O functions like fopen and fclose. Note that you must provide your own buffering for the low-level functions.
- If you do your own I/O buffering, make the buffer a multiple of 4K.
- If you are using a file as a temporary file and performing frequent read or write operations on it, use memory files. Because memory files operate on the memory of the system, I/O operations can be performed more quickly on memory files than on disk files.
- Instead of scanf and fscanf, use gets or fgets to read in a string, and then use one of atoi, atol, atof, or _atold to convert it to the appropriate format.
- Use sprintf only for complex formatting. For simpler formatting, such as string concatenation, use a more specific string function.

Designing and Calling Functions

Whether you are writing a function or calling a library function, there are a few things you should keep in mind:

- Fully prototype all functions. A full prototype gives the compiler and optimizer complete information as to the types of the parameters. As a result, promotions from unwidened types to widened

types are not required and the compiler does not need to emit eyecatcher instructions for the function.

- When designing a function, place the most used parameters in the lexically left-most position in the function prototype. The left-most parameters have a better chance of being stored in a register.
- Avoid passing structures or unions as function parameters or returning a structure or union. Passing aggregates requires the compiler to copy and store many values. Pass or return a pointer to the structure or union instead.
- If near the end of your function, you call another function and pass it the same parameters that were passed to your function, put them in the same order in the function prototypes. The compiler can then reuse the storage that the parameters are in and does not have to generate code to reorder them.
- Use the intrinsic and built-in functions, which include string manipulation, floating-point, and trigonometric functions. Intrinsic functions require less overhead and are faster than a function call, and often allow the compiler to perform better optimization.
- Avoid using intrinsic functions in loops. Many intrinsic functions use multiple registers. Some of the registers are specific and cannot be changed. In the loop, the number of values to be placed in registers increases while the number of registers is limited. As a result, temporary values such as loop induction variables and results of intermediate calculations often cannot be stored in registers, which slows your program performance.
- Use recursion only where necessary. Because recursion involves building a stack frame, an iterative solution is always faster than a recursive one.

Other Coding Techniques

The following list describes other techniques you can use to improve performance:

- Minimize the use of external (extern) variables to reduce aliasing and improve optimization.
- Avoid taking the address of local variables. If you use a local variable as a temporary variable and must take its address, avoid reusing the temporary variable. Taking the address of a local variable inhibits optimizations that would otherwise be done on calculations involving that variable.

- Avoid using short int values, except in aggregates. Because all integer arithmetic is done on long values, using short values causes extra conversions to be performed.
- If you do division or modulo by a divisor that is a power of 2, make the divisor unsigned to produce better code.
- Use #pragma alloc_text and #pragma data_seg to group code and data respectively, to improve the locality of reference. Variables that are used at the same time are stored together, and might fit on a single page.
- Use optlink linkage wherever possible. Make optlink your default linkage (it is by default) and use linkage keywords to change the linkage for specific functions that need to be _system linkage.
- If a loop body has a constant number of iterations, use constants in the loop condition to improve optimization. For example, for (i= 0; i< 4; i+ +) can be better optimized than for (i= 0; i< x; i+ +).

Certain coding practices will slow down your performance. Only use them if you need to. They are often necessary, but you should be aware that they will affect your program's performance:

- Calling 16-bit code. The compiler performs a number of conversions to allow interaction between 32-bit and 16-bit code.
- Using the setjmp and longjmp functions. These functions involve storing and restoring the state of the thread.
- Using #pragma handler. This #pragma causes code to be generated to register and deregister an exception handler for a function.
- Using variable argument functions. Due to the nature of optlink linkage, variable-length argument lists make performance much slower. If you use variable argument functions, use system linkage.

That's it for now on performance. Now let's take a look at .EXE size.

Putting your EXE on a diet

We're hearing from some people that their work with C Set/2 results in a big fat .EXE, and some hints would be appreciated on how to reduce the size of such files. Bear in mind, performance can be affected by determined efforts to slim your .EXE. Only you can decide if the tradeoff is worth it.

Choosing Compiler Options

The following list gives you compiler options which will help in size reduction:

/Gd+ Links dynamically to the runtime library. If you link statically, code for all the runtime functions you call is included in your executable module.

/Gf+ Generates code for fast floating-point execution and eliminates certain conversions.

Note: Code produced using /Gf+ does not conform to ANSI or IEEE standards.

/Gh- Does not generate profiler hooks which would increase module size.

/Gw- Does not generate an FWAIT instruction after each floating-point load instruction.

/G3 Optimizes for the 386 processor. Optimizing for the 486 generates extra code. Code compiled with /G3 also runs on a 486 processor.

/O+ Turns on optimization.

/Sh- Does not include ddname support.

/Ti- Does not generate debug information which would increase module size.

/Ts- Does not generate code to allow the debugger to maintain the call stack, which would increase module size.

If you link your program in a separate link step, specify the /ALIGN:1 linker option to align segments on 1-byte boundaries. The default alignment is 4-byte boundaries.

Using Libraries and Library Functions

Your choice of libraries and of library functions affect the size of your code:

- Use the subsystem library whenever possible. This library has no runtime environment, meaning the initialization, termination, and exception handling code is not included.
- Use the low-level I/O functions. Note that you must provide your own buffering for these functions.
- Disable the intrinsic functions. Certain string manipulation, floating-point, and trigonometric functions are inlined by default. To disable the inlining, parenthesize the function call, for example,

```
(strlen)("ian");
```

Other Coding Techniques

The following offers other ways you can use to make your modules smaller:

- If you do not use the argc and argv arguments to main, create a dummy _setuparg function that contains no code.
- Avoid assigning structures. Instead, use memcpy to copy the structure.
- Use #pragma strings(readonly) to make your strings read-only.

We hope you've seen enough in these two articles to help you become faster and slimmer. Our thanks to the many readers who suggested these topics as subjects.

16 going on 32...

While we'd all like to write pure 32-bit applications that exploit the true power of OS/2 and C Set/2, there are many times that we have to use 16-bit code in our 32-bit applications. The application may use a subsystem (e.g. DBM), or vendor library, which is still 16-bit code. Also, mixed 16-bit and 32-bit applications may occur while migrating code from 16-bit to 32-bit. Whichever, 16-bit isn't going to go away immediately.

C Set/2 provides many language and library features to enable mixed- mode programming. However, some of our users find mixed-mode programming somewhat difficult, so over the next few issues we'll try to shed some light on the subject. This month, some basics:

What you can do

- call 16-bit functions from 32-bit functions
- call 32-bit functions from 16-bit functions (try that with other compilers!)
- pass data between 32-bit and 16-bit functions
- share data between 32-bit and 16-bit modules

Calling 16-bit functions

What type of 16-bit functions are callable from C Set/2?

Three 16-bit linkage conventions are supported by C Set/2:

- C linkage - the standard C convention, parameters pushed in right to left lexical order, caller cleans up the parameters
- Pascal linkage - parameters pushed in left to right lexical order, callee function cleans up parameters
- fastcall linkage - the Microsoft C 6.0 register linkage convention

You can find all the picky little details of these three 16-bit linkage conventions in the C Set/2 User's Guide.

How do you declare a function as 16-bit?

You have two methods of declaring a linkage convention:

- #pragma linkage
  - #pragma linkage(function_name, far16 cdecl)
  - #pragma linkage(function_name, far16 pascal)
  - #pragma linkage(function_name, far16 fastcall)

  Note that the cdecl/pascal/fastcall linkage specifiers may optionally be prefixed with an underscore.

  The linkage pragma can be applied to individual function declarations, or to typedef's:

```
int maxine(int);
#pragma linkage(maxine, far16 pascal)
```

  or equivalently:

```
typedef int func16(int);
#pragma linkage(func16, far16 pascal)
func16 maxine;
```

  Unfortunately, you can't apply a linkage pragma to a function pointer. To set the linkage on a function pointer using a pragma, you must declare a typedef, apply the linkage pragma to the typedef and then use the typedef to declare the pointer. Admittedly, this can be inconvenient (and can lead to a heap of typedef's that only get used once), so your C Set/2 development

crew recommends that you use the second method of setting function linkage:

- linkage keywords
  - _Far16 _Cdecl
  - _Far16 _Pascal
  - _Far16 _Fastcall

  Linkage keywords can be used in any function or function pointer declarator. However, there is one common trap you should avoid. Many C compilers bind linkage keywords to the right. This mean the keywords apply to the objects to their right. Thus, in the declaration

```
int far cdecl *philip;
```

  the linkage keywords modify the pointer, philip. However, in ANSI C, type qualifiers such as linkage keywords bind to the left. C Set/2 has chosen to follow the ANSI direction in this question, so the above example should be coded as:

```
int * _Far16 _Cdecl philip;
```

How do you control the size of the 16-bit stack?

By default, any C Set/2 program that makes 16-bit calls receives a 4KB 16-bit stack for use by the called 16-bit functions. The size of this stack can be changed with:

```
#pragma stack16(stacksize)
```

where stacksize is the size of the stack in bytes, limited to a maximum of 65532 bytes. Please remember that the 16-bit stack is allocated from your program's 32-bit stack, so if you increase the size of your 16-bit stack, you may need to increase the size of the 32-bit stack. Also note that the parameters to the 16-bit function being called will occupy space on the 16-bit stack.

Linking with your 16-bit code

Contrary to popular opinion it is possible to statically link 32-bit code generated by C Set/2 with your favourite 16-bit object modules. However, there are many, many restrictions:

- main must be in the 32-bit code

- you can't use any C library calls in the 16-bit code (we haven't done any work to make the C Set/2 library coexist with 16-bit C libraries)
- the 16-bit code must be compiled with the /ND option (the linker can't tolerate having both a 32-bit and a 16-bit DGROUP)

The above restrictions really do make statically linking 16-bit and 32-bit code impractical in most cases. There are no restrictions when linking a 32-bit EXE to a 16-bit DLL. But don't forget, that when exporting 16-bit function names in a DEF file that the correct translation must be made to the names:

- _Far16 _Cdecl - add underscore (_) prefix, i.e. fred is _fred
- _Far16 _Pascal - uppercase the name, i.e. fred is FRED
- _Far16 _Fastcall - add at-sign (@), i.e. fred is @fred

Coming Soon

Well, that's all the bits we can squeeze in for now. In future issues, we'll discuss 16-bit code calling 32-bit functions, and passing, and sharing data between 16-bit and 32-bit code. We'll also take a peek below the covers at what happens when a 16-bit call is made, and discuss calling some of the ES subsystems.

Migration hint

When migrating 16 bit code into C Set/2, pay attention to how you migrate pointer parameters to the APIs. Given that your 16 bit API was called in the following way:

```
USHORT x;

DosXxxxx(..., &x , ...)
```

Don't migrate it by changing the API call to:

```
DosXxxxx(..., (PULONG) &x , ...)
```

Migrate it by changing the variable definition to:

```
ULONG x;
```

Under some circumstances, using the cast appears to work. However, this is incorrect code, since the API will write back 4 bytes, not 2. Whether you have a bug will depend on what variable is stored in the two other bytes. Since the layout of automatic variable can change when you optimize your code or add debug information, the bug may disappear or change characteristics when you change the /O and /Ti compile options.

You ask, we answer

These questions are culled from a variety of sources: from internal and external fora; from customer calls; and from the reply forms mailed in by our readers.

Question	Answer
Q: WorkFrame/2 and Edge Editor I just installed the EDGE Editor under the WorkFrame. Is possible to have the compiler errors in EDGE as well as with EPM and the /W switch ? I tried /W %a %z, but it doesn't seem to work.	The only way to have other editors 'know' about compiler errors from the WorkFrame/2 is if they support the WorkFrame/2 DDE interface. For EPM, this is turned on (in the editor) by using /W and in the WorkFrame by selecting the checkbox in the Configure Editor dialog. Bottom line...EDGE has to add support for the WorkFrame.
Q: Closing action log The online help says I can't close the action log window. However, I can close the project control window. Is there any reason to have one closable and not the other?	The Action Log is not closable. We suggest you minimize it out of the way if you don't need it.
Q: Full Screen Tools. I am trying to use an application as a tool that uses a Full Screen. Once the tool terminates, WF removes the full screen and the results are gone. Is there any way I can hold the screen until I am ready to return to WF? (I am using WF on OS2 1.3 EE).	That's an easy one. Instead of calling the application directly, call cmd.exe instead and give the parameters '/fs /c appname.exe options'.
Q: Capture of VIEW.EXE not working I have "Capture Tools" turned on in my preferences but it doesn't seem to work either when putting VIEW.EXE on the tools menu or by using the Ctrl-H method of getting help. The help window goes under the WF/2 window and out of sight. The Seek/Scan program from the Productivity folder does just what I want it to do when I load it from the Tools menu - stays around in WF/2 in zorder.	VIEW is a strange program in that it starts another program to actually display the help and then exits immediately. So the behaviour that you are seeing is perfectly normal. I can't even suggest a workaround. Sorry.
Q: Debug (IPMD) parameters at call from WF. Each time I run Debug from WF pulldown, I have to reapply my program's parameters. I expect these entered in Run Options to be passed to IMPD. Am I doing something wrong ?	Did you select the 'save with project' checkbox when you entered the options OR are you entering the options only when selecting debug and not by selecting the Debug entry under the Options pulldown? Use the %r option in the debug parms.

Question	Answer
Q: IPMD command line parameters This is just a nit, but it compels me to change the startup parameters quite frequently. Changing the startup parameters means the application is started, stopped and restarted. My command: START IMPD TESTPROG "a string which is one parameter" The program under test is given a parameter for each word in the quoted string. I would imagine redirected files would also behave similarly, i.e. the debugged program's files would not be redirected. Redirected files are used extensively for trace logs in PM programs.	What you are seeing is a restriction which occurs when you include the name of the program to be debugged and the parameters on the invocation line with IPMD. If you don't have a requirement to pass this information on the command line, the following (from the read.me) may help you: "When the debugger is invoked from the command line, or from the the WorkFrame/2 product, with parameters containing strings in double quotation marks, the quotation marks are not preserved. To pass quoted strings to the application, you must use the debugger startup dialog."
Q: IPMD Documentation Is there a User's Guide or like for IPMD? If so, where can I get it?	There is a "C Set/2 Debugger Tutorial", S10G-4447.
Q: Data representation If I have a e.g. void *, is there any possibility to "cast" it to another data type pointer during debugging to force the debugger to use this data representation (especially if the symbolic information) is provided ? Think of a "general purpose" pointer which is used to address different data objects (one at a time) and therefore was declared PVOID and is explicitly casted (e.g. (PFILEFINDBUF3)pfoo) when needed. I think under Codeview it was possible to add (PFILEFINDBUF3)pfoo to the monitor and the representation was "pointer to FILEFINDBUF3". Is there any magic to force IPMD to handle this situation accordingly ?	At this time there is no way to cast a pointer to a pointer to another data type. This support will probably be in the next release of the debugger.
Q: Debugging a DLL which has not yet been loaded! I am trying to debug a DLL which is dynamically loaded. Since I don't have the source code of the main program I can't even set a breakpoint immediately after the place where the particular DLL was loaded. With the CVP I used the /L option. Which means do I have with the IPMD debugger?	You can set a breakpoint which takes when your DLL is loaded: use the Breakpoints menu, drop down the list in the top left and select Load, and put the name of your DLL as the parameter, with the .DLL extension on the end. You'll hit the breakpoint when the DLL is loaded, and the Program Parts list will include your code.

Question	Answer
Q: IPMD problems I get the following error when trying to use IPMD: Program not loaded. DosStartSession failed (rc= 127) Does anyone have any ideas???	127 normally indicates a linkage problem - it found the dll but couldn't find one of the exported entry points. Either a) you've built a bad dll or b) you're picking up a dll that's out of sync with what ever is trying to call it.
Q: Printing annotated listing Once I have opened up the annotated listing, is there a way to print all or part of that listing ??	There are no print functions available from within the debugger.
Q: Dumping memory I was wondering how to simply dump memory at a given address in the PM Debugger? I didn't see it anywhere in the documentation.	CTRL-G for a new storage window and then you can see memory in any format that you want.
Q: PMWIN In the Source window I get stuff like PMWIN:8 etc...I'm presuming the 8 (in this case) is the ordinal of the function in PMWIN.DLL. Am I correct in presuming this? If not, could somebody please let me know what that number represents.	The number following the entry is an index which IPMD assigns to the memory objects contained in the DLL or EXE. If the DLL contains 4 memory objects, IPMD numbers them 1 through 4 and uses that number to reference them.
Q: Changing 'Project Control' directory How do I tell WorkFrame to use a different 'Project Control' directory. I would like it to show me a limited number of projects, not just whatever *.prj files it finds in \IBMWF. Also, it would be nice if the 'Project Control' window listed projected by alpha order (and I don't mean alpha order by .prj file name!).	If you're really eager to do this, you'll have to get some kind of OS2.INI editor and change the value of the IBMWF/DIR application/key pair to point to a different directory. You can only do this between invocations though. A better method is to use our ADDUSER tool. It will update the .INI file AND copy the other relevant files needed for the WF to start up. The previous method would require you to copy the various .INI, .PRF and .BMP files. Don't select to copy the .PRJ files. Copy the ones you want in the new list afterwards. This method should be easier. We'll try to add selection for release 2. We'll take a look at the sort order in release 2 too.
Q: Is there a list of the errno values returned by C-Set/2 calls? I have looked at some of the C-Set/2 docs, and I can't seem to find a list of the errno values that C-Set/2 functions return upon failure. Is this info available? If so, where is it?	Each specific function description has a section called "Return Value", which says what the function returns (obviously) and what the possible errno values are, if any. Also in Appendix D (Run-Time Messages) where the run-time messages are listed, the errno value associated with each message is also listed.

Question	Answer
Q: Defaults for C Set/2 I am trying to change some defaults settings in the WorkFrame for C-Set/2, such as compile with no link and set debugging information, so when I create a new project, these settings will already have been made. Is this possible in the Workframe?	You can copy options from an existing project to a new project. This is not as nice as the ability to set "global" options would be, but it is often much more convenient than creating new projects from scratch. When you create a new project, and the New Project window is presented, you can drag a project from the Project Control window to the Project Description field in the New Project window. This will copy the existing project to your new project, and you need only change the options that are different between the old and new projects. In particular, make sure the Link options...Filename options... are correct; usually they won't be correct because they will correspond to the old project. You can take this method a little further by having a 'default C Set/2 project' setup instead of copying an existing project. In the default project, you can specify some of the parameters in such a way so that they will force a failure if they are not changed. This will alert you to that fact if you do forget to change the options.
Q: Composite Program under WF won't let me enter I am moving a large project into the workframe environment. It has about a dozen objs (one created with 32 bit assembler) that eventually get linked. I think I should make this a composite project, so I quickly made up a dozen or so base projects that had the names of most of the obj files. Then I tried to form a composite project. It let me add in the names of the base projects, but would not highlight the ENTER button, so all I could do was cancel, which erased everything. The help just said to enter the name of a project before I hit the add button (which needless to say was of no use at all). The tutorial that comes with WF doesn't have a composite project as an example.....what do I do?	You need to highlight one of the subprojects first--this will be your 'run' project.
Q: Editor questions Why can't the "editor" be a .cmd file? And what directory is "current" in the process in which the editor is launched? I seem to get the root directory by default (this might be due to the same problem that causes %d to be null).	The editor must be an executable file. If you want to run a .CMD file, specify CMD.EXE as the editor, and then /C cmdname.CMD as the parameters.

Question	Answer

Q: PRINTF failing when using "_Seg16"
 	
In this code, why does the first "printf" gives a Protection Fault Exception?
 	
`#include < os2.h>`
`#include < stdlib.h>`
`#include < stdio.h>`
`#include < string.h>`
 	
`VOID main( )`
`{`
`PCHAR _Seg16 SPtr ;`
`CHAR string(50) ; /* Yes, they are really SQUARE`
`int ch ; /* brackets.`
 	
`strcpy( string, "testthisdata") ;`
`SPtr = string ;`
 	
`printf("%s\n",SPtr) ; /* This is the failing statement`
 	
`printf( string ) ;`
 	
`ch = getc( stdin) ;`
 	
`}`
 	
`---------- `
`# IBM Developer's Workframe/2 Make File Creation`
`run at 15:50:14 on 09/08/92`
 	
`# Make File Creation run in directory:`
`# C:\IBMC\WKFRAME\PROB44;`
 	
`FFIXES:`
 	
`FFIXES: .c`
 	
`prob44.exe: \`
`prob44.OBJ \`
`PROB44.MAK`
`@REM @< < PROB44.@0`
`/CO /M +`
`prob44.OBJ`
`prob44.exe`
`prob44.map`
 	
`;`
`< <`
`LINK386.EXE @PROB44.@0`
 	
`{.}.c.obj:`
`ICC.EXE /FoPROB44.obj /Ti /C .\$*.c`
 	
`!include PROB44.DEP`
	printf() is a varargs function which expects all pointer parameters to be 32-bit pointers, Sptr is a far 16-bit pointer so when printf() tries to use it, it goes BOOOOOMMMM!
	
	printf expects a 32-bit char pointer for the %s replacement. There is no way for the compiler to know which is expected. One solution would be to put a (char *) cast in front of the offending parameter to tell the compiler to cast the pointer into a flat address.

Q: Compiler options DLL How does one go about creating a compiler options DLL? I would like one to use with PLX/86.	If you have installed the sample projects that came with the WF, then in the subdirectory ...\PRJ\SAMPDLL, you will find the source code for the default options DLL. You can use this as a base for writing your own options DLL. Detailed documentation and samples are available from us.

Question	Answer

Q: Function Prototyping
 	
I am having trouble with a function prototype. I declared a function as MRESULT (* _Seg16 _Far16 _Pascal fn)(); Further down in my program, at the actual function, I have...
MRESULT (* _Seg16 _Far16 _Pascal fn)()
{
program statements
}
 	
When I compile this program, the compiler takes the function declaration but it complains about the declaration further down. The message I get is...The declaration or definition of the function is not valid. Can someone tell me what I am doing wrong?
	The first declaration declares fn which is a pointer to a function. The second tries to use that pointer to a function to declare a function and the compiler goes huh? and refuses to compile it. Get rid of the (* _Seg16 and) tokens and you'll have a function called fn that has 16-bit pascal linkage.

Q: Listing wish I love the listing capabilities of CSet/2: X-ref's structure layouts etc. But there is a slight problem... I've been trying to debug a structure in one of my headers. When I selected "Variables layout", the listing contained my structures AND all those in the OS2.H include hierarchy. I ended up with a 130 page listing for a 60 line program!!!! WIBNI if we could selectively turn-off the variables layout and Macro expansion for particular headers. I could say "no expansion for OS2.h" and get a radically reduced listing size! (Save a few trees, too!)	It's already on the list.
Q: Very annoying problem with very simple program While trying to compile a "hello world" type program with the command "icc filename.c" (in other words, without any options), I received this error: error EDC0637: Unable to open linker response file Y:\IBMC\TMP;Y:\edc00. Now, after spending a decent amount of time looking up the error, I found that the recovery for it was to check the amount of disk space. (I have plenty of disk space.) What sort of things can give you that error? Incidentally, I'm not using a response file - if ICC is, I'm not aware of it. Any help would be appreciated.	Your TMP env. var is probably set to Y:\IBMC\TMP;Y:\ so when it tries to create the linker response file, it generates an invalid name. Use only one directory on the TMP var.

Question	Answer
Q: Error compiling with Optimization turned on When compiling a file under CSET/2 with optimization turned on I get the following message : . F:\source\drv0\sqt‘icc /C+ /Ti+ /O+ /Lsi+ sqltb.c Licensed Materials - Property of IBM IBM C Set/2 Version 1.0 (C) Copyright IBM Corp. 1991, 1992. All Rights Reserved. US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp. Term signal exception received and executed . F:\source\drv0\sqt‘ Compiling the same file with optimization turned off I get no errors. Has this problem with CSET/2 been reported before ?	This is sometimes caused by not having enough room for your swapper.dat file on the disk. Turning on optimization uses more memory and therefore more disk (in swapping situations).
Q: Debug and Run are Grayed out in the Actions drop-down menu I compile my program successfully. However, I can't debug or run my program from the Actions drop-down menu because those options are still grayed out even though the compile completed successfully. What does this mean?	Did you link after you compiled? Is your target .exe present in the directory?
Yes I linked after I compiled. The EXE file has been created. I go to an OS/2 prompt and check the directory and the EXE file is there. The EXE file works up to a point and then it traps (SYS3171 - Insufficient stack). It looks like the fact that the program traps is related to the fact that WorkFrame grays out the Debug and Run options in the Actions drop-down menu. However, there have been no errors in the compile or link steps. Now if I could find out why they get grayed out maybe it could help me figure out why it is trapping.	For any number of reasons, the executable you are building has a different name than the executable (target) that you specified when creating the project. You may have left off the extension, or misspelled it as suggested previously.
Q: Terminating a thread using DosKillThread Can I terminate a secondary thread, created with _beginthread, using DosKillThread? Are there some necessary cleaning up that must be done?	It is not a good idea to terminate a thread with DosKillThread. As you have already indicated, there is cleaning up to do. If this clean-up is not done, you have two problems 1. wasted storage 2. new thread created may end up using the thread storage of the "killed" thread if the new thread has the same id as the old thread. This is not desirable since the initial state of the thread specific data are not known.

Question	Answer
Q: Problem adding a tool I added a tool using the Configure.Tool... choice and I can't get it to execute. When I select it from the Tools menu, I get an error saying that it can't be loaded and an OS/2 error code of 3. The help text says rc 3 means that the file could not be found. I didn't fully qualify the file at first. Other help test said that if I don't specify the path for the executable file, WorkFrame will search my defined path. The executable resides in a directory in my search path. What's more, if I do fully qualify the file, I get the same error. I also tried specifying the system editor (E.EXE) and EPM.EXE and had the same problem. This function seems too easy to screw up.....what am I doing wrong?	RC= 3 is not exactly file not found, it's path not found. Are you sure the startup directory is correct? (it must not be blank -- if you don't care, put "." there, to indicate the current directory).

Q: Compiler options
 	
I'm trying to set the compiler options from the OPTIONS menu. The default option is presented as % f OBJECT LIST. I would like to change it to % d % d % f OBJECT LIST. When I try this I get the message "only one instance of the symbol % f is allowed". I don't understand? I only have one instance of % f. If I change to % d % f OBJECT LIST I get the same message. If I remove the % f altogether - still the same message.
 	
What I'm trying to do is set up the AL/86 assembler as another language, which is invoked by ASM.CMD. I've successfully built an ini file and ADDTOOLed to WKFRAME. I can certainly invoke it from the TOOLS menu. I have built a language profile for it and would like to invoke it from the ACTIONS menu from compile.
	There is a restriction built into the default DLL to allow % f as the only compile option substitution parameter -- I'll agree that the error message is also misleading. I won't try to explain why it was designed that way , but as the source to that DLL is shipped with WorkFrame (in \IBMWF\PRJ\SAMPDLL), I can tell you how to fix it. (This assumes you have a 16 bit C compiler -- either IBM C/2 or MS C 6 will do).
	
	Edit metaver.c, and add the following to the switch() in comp_meta_string(), before the default:
	
	`case 'a':`
	`case 'z':`
	`case 'd':`
	`case 'e':`
	`case 'm':`
	`case 'n':`
	`case 'o':`
	`case 'r':`
	`break;`
	
	This will allow all substitution parameters to be used.

Q: Help not working Help does not work on MY machine! Clicking on the Help choice leads to the message 'IPF could not be initialized'. What does this mean? What do I have to do to get it running?	The most common cause of this is not having the following in your config.sys: SET HELP= C:\IBMC\HELP;...

Question	Answer
Q: Workframe/2 and the OS/2 PATH statement. I have a number of projects setup in Workframe/2 and each one of them uses a different compiler. I actively switch between projects that require IBMC-32bit, IBMC2-16bit, MSC6.0 and Zortech C+ + 3.0. The problem I am encountering is I have to add all the possible executable directories into my PATH statement and reboot my machine so that Workframe/2 can pickup the appropriate PATH to find the required files to successfully make the executables. I would like to specify the search path for each language profile in the language variable setup. It is as if Workframe/2 has fallen just a little short of providing the prefect multiple language environment. I do not want to have all my compilers and required project executable directories specified in my PATH statement and I shouldn't need to if Workframe/2 would provide the facility to search a language specific PATH statement instead of the system PATH. There are many conflicts that occur and can occur when all my compilers directories are included in your PATH. Is there a way to fix my problem without the need to modify my PATH and reboot so that Workframe/2 can recognize my changes?	You can specify directories to be pre-pended to the LIB and INCLUDE paths, but for version 1, you must add all the compilers to your path statement in the CONFIG.SYS. We are enhancing the environment variable support for the next version.
Q: Linking files I have been trying to use WorkFrame/2 to build a small PM program. One of the things that I specified under the linker file names was a .def file. If I do a compile on the source C file using C-SET2 it does a link in addition to the compile. However, the .def file name is not passed to the linker. The link works fine as does the following RC but the executable fails with a 1051 error code - not in a PM session. If I execute only the linker I can make it take the .def file definition. Other options passed to the linker are different as well. I am obviously missing something here, but what?	You're linking with the compiler, whereas it should be a separate step (using LINK386). Select 'Compile only' from the compiler-> other options. The compiler can do the linking, but the WF/2 options dialog does not allow you to set any of the link options. To use a def file, type the filename in the link-> filenames options, and link as a separate step.

Question	Answer
Q: Creating a MAKEFILE for an .EXE and .DLL I am developing a program that has both an .EXE and a .DLL in it. The source is in separate PROGA.C and PROGADLL.C. How can I use the MAKEFILE Creation tool to create a MAKEFILE for both the .DLL and .EXE files?	A base project can only have a single target, and a single set of options for LINK, COMPILE, etc (different options would be required to build a DLL and EXE). You could set up two base projects, one for the DLL, and one for the EXE (both can be in the same directory if you wish). You can then create a composite project, containing both base projects, which will allow you to build both at the same time.
Q: Returning errors to editor Help!! I've searched high and low for help on how to setup the Error Message Template and can't find it. I've seen mention of the Compile Options dialog box but none of the choices found in this box have a place to put the template. I'm trying to use the C Set/2 compiler with the Enhanced editor.	You don't need to set the error msg template -- that's only provided so you can use compilers that are not WorkFrame enabled. If you're using C Set/2 (which is WorkFrame enabled), the compiler DLL already knows how to parse its own error messages correctly. Don't forget that to see the error messages from within EPM you need to insert /W at the beginning of the editor invocation string (from configure editor).
Q: Feedback on Workframe A minor observation about Workframe/2. With regard to Workframe I believe there is a small design flaw. The error message template used to tell Workframe how to look up error messages appears as a compile option the project window. Surely this format varies from compiler to compiler, and not project to project, and so should be part of Language Profile Management.	While that idea sounds good in isolation, the current design is valid. Each compiler used within WorkFrame requires a compiler options DLL, that provides the ability to set compiler options, and also to parse its own error messages. The compiler options must obviously be settable on a project basis. The error message parsing is specific to that compiler, and so is normally hardcoded into the DLL (and does not require any options). Given that framework, it was then decided to provide a default DLL, that could be used by non-WorkFrame enabled compilers. The error template is an option unique to this DLL, and it would not really make sense to add it as an option for all compilers. The obvious compromise is to save the template as one of the compiler options (which is what's done), and simply display it as a separate item, from the compiler options pulldown. Please note that if you move to a WorkFrame enabled compiler, the "error template" option will go away completely. Hope that helps.

Question	Answer
Q: WorkFrame/2 Beginner Question I am new to OS/2 programming, though not to using OS/2. Yesterday I installed the OS/2 2.0 Developer's Toolkit, WorkFrame/2, and C Set/2 on a PS/2 Model 80 running OS/2 2.0 (GA level). I allowed each of the programs to modify my CONFIG.SYS as it saw fit. Next I tried to step through the HELLO2 example in the WorkFrame/2 Integrated Development Environment document. Everything worked fine up to the point where I ran the compiler and linker. First time, the compiler told me it couldn't find STDIO.H. Well, after some searching, I found the location of the file and created a statement in my CONFIG SET INCLUDE= C:\IBMC\ This fixed the compiler error. I would have thought that this statement would have been added automatically when I installed the compiler. Why not? The second problem I have not been able to resolve. After getting the compile to work, the link fails. The message is: LINK386: warning L4051 : OS2386.LIB : cannot find library Enter new file spec:LINK386 : warning L4051: dde4sbs.lib : cannot find lib followed by several L2029 errors (unresolved external) Though I see that dde4sbs.lib is in the subdirectory C:\IBMC\LIB, adding that directory to my LIBPATH seems to make no difference. What do I need to do to get this working? And again, why didn't the installation program take care of this automatically?	Both those lines (BTW, the LIB files should be listed on a LIB path, not a LIBPATH), should be added to your config.sys file when you install the C Set/2 compiler. If, when you install C Set/2, you elect not to update the config.sys, a CSET2ENV.CMD file will be created that sets the environment up correctly for you. In this case, you'll need to execute that file from the same session, and before starting WorkFrame. Edit the WFENV.CMD file (created when you install WorkFrame) and add the line CALL CSET2ENV to the beginning. This will automatically set up your environment.
Q: Pascal/2. Can WorkFrame work with Pascal/2? I created a language profile for IBM Pascal/2 but when invoking the compiler I get the following in the compile window: invocation of < pascal.exe> begins invocation string: TEST.PAS < ---------->< Compile - TEST.PAS> ended tc= 0 rc= 8 at... What does rc= 8 mean? The compiler is not even executed.	The path to the Pascal compiler must be in the first 132 characters of PATH. When WFENV.CMD changed PATH, it moved the compiler's path beyond 132 chars.

Question	Answer
Q: Passing /I How do you pass the /I parm from WKFRAME to the debugger? I changed the Debug options to include this with no luck.	You just select 'debug' under the 'options' pull-down. The ordering of the parameters you pass should be something like: 1. options to IPMD.exe 2. name of executable to debug (or %o/%f/...) 3. parameters to executable You could also not put any options at all and then PMD would put up an initialization dialog to get the required information. Parameters for IPMD are: • /i start debugging with application at the OS/2 initialization of the application. This will allow one to debug the initialization of global variables. Note : when main() is reached this initialization is finished. The default is to start debugging at main(). • /n Do not use the restart information. As much restart information is saved as possible. Open windows, breakpoints,... Note: only thread 1 information is saved since other threads do not exist when the IPMD is started at main(). The default is to use restart information. This information is in the Debugger Tutorial, page 2 "Starting the Debugger from the Operating System/2 Command Prompt".
Q: Invoking EPM from monitor box When I invoke EPM by double-clicking on an error message line in the monitor box, the editor is invoked with the file containing the error, but I do not get the highlighted lines or the "Compiler" selection on the action bar as I do when I drag the error onto an already open edit window. Also, when the editor is invoked in this way, the drag drop interface from the monitor box appears to be disabled. Can someone please tell me what I am doing wrong? I am using the LA toolkit and I did select the "SendCompiler Errors" checkbox in the Configure Editor dialog.	Add a /W to the front of the editor options string. For eg., if you had '%a %z' make it '/W %a %z'. If you're using a non-WorkFrame-enabled compiler and the default compiler DLL (DDE3DEFL), make sure that the error template is correct.

Question	Answer
Q: Adding 'iconedit' as a tool I tried adding iconedit.exe as a tool, but I keep being told it couldn't load the file (rc= 3). What do I need to do in order to add this tool?	Say addtool D:\toolkt20\toolkt20.ini where D: is the drive where your toolkit is installed. That should install iconedit, dlgedit, and fontedit.
Q: NMAKE error message U1077 C\OS2\CMD.EXE return code 17 Any ideas on the meaning/cause of error message NMAKE : fatal error U1077 : 'C'\OS2\CMD.EXE' : return code '17' The samples work fine - I copy a sample to a new subdirectory, set up the project etc, no error, then when I execute make the above message results. Any and all suggestions appreciated.	The make file that you were processing probably contains the command rename or copy which is no longer valid from the new subdirectory. Add a -d option to the nmake options and you should be able to see which command is failing.
Q: Use of full screen editor as Workframe editor I'd like to use a fullscreen editor as the Workframe editor, but I run into two problems: 1. it always gets executed in an OS/2 window; I can't find a way to say that I want OS/2 fullscreen 2. if I try to specify the editor as a command file (e.g EDIT.CMD) - because I want to set MODE - the Workframe says it can't find it Is there a way around these?	• The editor will always be started in a window if possible. To start it from a full screen via a .CMD file, use configure/editor dialog. Input "cmd.exe" as the editor, and "/C start /fs editor.exe" as the invocation string. You can specify the editor as an OS/2 Full Screen tool using the configure/tools dialog, but this will only be used when you select edit from the tools menu, not when you double click on a file. • Use configure/tools/new to create a new tool. Specify edit.cmd as filename and then click "Command File" check box. Finally, enter % a % z as the parameters to pass. This will cause all selected files to be edited when Tools> Edit is chosen.
Q: Having the Make File Generator Pick up header files Why doesn't the WorkFrame/2 make file generator pick up .H and .HPP files as dependencies when I "click" on them? It seems to find them for the resource compiler (rc), but not the main compiler... What do I need to change in my set up to enable this to happen?	It appears that you are using the default compiler DLL. Assuming that you are using the C Set/2 compiler, make sure that the compiler DLL (in the language profile for C Set/2) is set to DDE4CSET.DLL, not DDE3DEFL.DLL.

Question	Answer
Q: Multithreading DLL (or not) Can someone tell me more about multithread / single thread run time coexistence?	If you have a DLL which you wish to use with both single and multithreaded apps, you **must** create it as multithreaded. You should also observe the following requirements: 1. Files opened inside the DLL may not be manipulated (read, written, closed, etc) outside the DLL, and vice versa. Doing so can cause some really hard to find bugs. 2. Memory allocated using the runtime library (maclloc, etc) inside the DLL must be freed/reallocated inside the DLL, and vice versa. Note that once the memory is allocated, both the DLL and EXE may use it. The limitation applies only to allocation and freeing. 3. You should specify #pragma handler() for all your DLL entry points. Failing to do so may generate unpredictable behaviour of signals, an the floating point math library. 4. You must assume that the library environment inside the DLL is different from that outside the DLL. This is not always the case, but if the DLL is linked to a single threaded EXE, it is guaranteed to be so. This affects functions like signal(), raise(), _fcloseall() strtok(), rand(), which keep or use data in the library environment. it also affects the behaviour of errno. To guarantee that this is always the case, statically link the libraries to the DLL. This covers the main points. If I were trying to do this, I would statically link the library to the DLL, and treat the DLL as an independent C runtime environment (with the occasional awkwardness this can entail)
Q: Exception handling Is there an easier way of installing the C-Set/2 exception handlers than putting in #pragma handler statements after every function that I export as a DLL entry point? Although it is not hard to do so, it does require that I modify a lot of files that I have not had to modify in my port to C-Set/2.	Exception handling is driven by the OS/2 model. It requires registration of exception handlers in the stack, hence the #pragma handler() statements. If you don't care that a given entry point has its exceptions handled by its caller, leave it out. Note that you then cannot use signal() or any function in math.h anywhere in the code path from that entry point (they're broken if the exception handler isn't registered).

Question	Answer
Q: Message file What message file do I need to BIND with an EXE file (obtained with IBM C-Set/2) to avoid "Message file not found" when the program is executing in a machine without the compiler installed ? I also need to know the number of messages to create the infile needed to run MSGBIND.	Bind in DDE4.MSG. Use the Workframe/2 Make file generation utility to generate a message bind step. It will give you the message numbers.
Q: Underflow exception How can we bypass this error. I want it so that any underflow result will be returned as a Zero. (like in Unix XLC).	Use the _control87() library function to set up the 80387 controls to mask off underflow.
Q: Unresolved Externals I am getting the following unresolved externals: error L2029: 'getch' : unresolved external error L2029: 'ltoa' : unresolved external error L2029: 'flushall' : unresolved external error L2029: 'kbhit' : unresolved external What do I need to link in? I have tried number LIB files, but no-go.	These functions are in the migration libraries. Compile with the /Sm switch, and the migration libraries are brought in automatically. I presume that you compiled with /Sm else you would've gotten compilation errors. Did you install the Migration Libraries? If you installed the default libraries then you did not install the Migration libraries. DDE4SBM.LIB - statically-bound, single-thread migration library DDE4MBM.LIB - statically-bound, multi-thread migration library DDE4MBMI.LIB - dynamically-bound, mutithread migration import library DDE4SBMI.LIB - dynamically-bound, sin-gle-thread migration import library

Q: Problems using printf and scanf
 	
I'm using C Set/2 for the first time and having problems with something I thought should work. My code has a number of printf/scanf pairs as listed below:
printf("\n\nEnter the number of months: ");
scanf("%d", &no_months);
printf("\nEnter the principal amount: ");
scanf("%f", &principal);
...
 	
In each printf/scanf pair, the scanf is occurring first, and then is followed by the printf (which is supposed to be the input prompt). If I put an extra \n at the END of every printf, then every-thing works OK. Is this a known feature/restriction/bug?
	The stdout stream is LINE BUFFERED. This means that output is sent to the real device when one of the following occurs:
	
	1. The library buffer is full
	2. The library detects a \n character in the output stream
	3. The user codes fflush(stdout)
	4. The user codes fclose(stdout)
	
	You look like you want UNBUFFERED. Use the library call setvbuf().

Question	Answer
Q: Softcopy of Manuals Where I can get the C SET/2 manuals in BookManager format? The hardcopy manuals reference *.BOO files, but I have been unable to locate them in any of the IBMBOOK disks. I searched by file name and file description with no luck.	The .BOO files are in the HELP subdirectory of the compiler (assuming that you installed them.....) - they're shipped with the C Set/2. You can install them with the installation program (look under the Documentation heading). The books provided in BookManager format are the User's Guide, Migration Guide, and SAA CPI C Reference. You can also look for a CD-ROM of OS/2-type books sometime this year.
Q: IBM QuickHelp for OS/2 2.0 Does there exist or is there in plan an easy-to-use C and API (Win, Gpi, Dos) reference guide similar to MicroSoft's QuickHelp for OS/2 2.0? With MS QuickHelp I can press ALT+ Q on any API listed within my code (while in my editor) and a screen will pop-up with listing of the specific API with all its parameters, example of usage, etc. Will also pull up descriptions of C syntax and error messages.	Yes - If you've installed WORKFRAME/2, the version of EPM that comes with OS/2 2.0 will do this if you hit Ctrl-H when the cursor is on an API name or C keyword.
Q: Questions on Compile Options 1. Are there any implications if I compile some files with /Sa and some with /Sm, then link them into a single executable ? 2. What happens if I compile with /C+ /Ge+ and link the object file into a DLL ? 3. Conversely, what happens if I compile with /C+ /Ge- and link the object file into an EXE ?	1. You can mix modules compiled with /Sa, /Se, /S2, or /Sm without restriction. 2. You can put a module compiled with /Ge+ into a DLL without problems as long as the function does not define the function 'main()'. There must be at least one module in the link that was compiled with /Ge-, though. 3. You may not put a module compiled with /Ge- into an EXE.
Q: Calling 32-bit appl from 16-bit appl As a newcomer to the 32-bit world I need to find out if we can develop an 32-bit application which will be called from an 16-bit application. My application in turn needs to call a 16-bit application. Any suggestions, comments and pointers to discussions about this problem are highly appreciated.	This shouldn't be a problem. You will have to put the 32 bit APP into its own DLL, and insure that its entry points are defined as 16 bit entry points, and the call backs to 16 bit are properly prototyped. Note that there can be some problems with pointers, since 16 and 32 bit code use different pointer formats. Proper declaration of the pointers will usually make things work. Please read the User's Guide, Chapter 16.

Question	Answer
Q: Multi-DLL runtime library considerations We have some code which consists of an .EXE and a fair number of .DLLs The number of .DLLs may be as low as 10, and I currently have 38 that eventually get loaded into the application. I'm wondering about the runtime library. If each .DLL and the .EXE have their own runtime library, what kind of overhead are we talking about in terms of size and performance (runtime library initialization and termination). Do multiple runtime libraries 'talk' to each other correctly so that I can open a file in one and use it in another? How about allocating memory in one and freeing it in another? (I'm talking the C runtime functions, not the OS/2 native calls). I'm assuming that it would be in our best interest to provide a runtime library with the code that each of the DLLs used. Is there any good reason not to, besides the fact that we have to remember to rebuild this runtime library every time the compiler gets updated?	There's some very good reasons to build your own library DLL: 1. With 10 DLLs, you could be talking major improvements in total size (both working set and on disk) and load time. Any reasonable DLL or EXE is going to bring in 20+ K of library code. A DLL library is going to be smaller than that by quite a margin. 2. You will notice an improvement in your build time. 3. Since the separate statically linked libraries don't talk to one another, it can make passing file pointers around tricky at best. It also requires you to free memory in the same module that you allocated it in. Both these restrictions disappear if a library DLL is used. N.B. I'd recommend linking the library into one of your own DLL's, and exporting the entry points from there.
Q: heapmin If I have multiple DLLs that each have their own version of the runtime, will _heapmin() clean up all the malloc/free memory blocks for all the DLLs, or only the malloc/free memory blocks in the DLL it was called from?	If you have multiple copies of the runtime (i.e. statically linked to each DLL and the EXE), then each copy of the runtime maintains its own environment (including the heap). Therefore, calling _heapmin() only minimizes the heap for one environment. We recommend that you use only one copy of the library, statically linked to one of your DLLs, and export its entry points to everyone else. You get smaller code, and faster load times.
Q: SYS3175 after exiting program I am working on a multi-threaded program. It runs fine until I exit the program after running several threads. Then I get a SYS3175 Access Violation. When I run under IPMD, I get thru the Program Ended popup, and then I get the same SYS3175. I know it is my problem; but how does one figure out where they are? The SYS3175 message says it is at location 0002A741, and shows me a bunch of registers. But how can I determine where 2A741 is, and what is happening at that location?	You have the information you need - the location of the trap. Create a MAP file when you link (a full map). You can then use that to determine the routine in which you trapped. Remember that EXEs are loaded at location 010000, and you're off to the races....

Question	Answer
Q.OS2386.LIB I need to use OS2386.LIB but I cannot locate it. Where does it come from?	This is part of the Toolkit. It installs by default into the TOOLKT20/OS2LIB subdirectory. If you install the Toolkit the install program should automatically update your CONFIG.SYS to include this directory in the search dirs for LIBs. If you don't have the Toolkit, compile with the /Gd+ option. The Link step for the library DLL's doesn't require the OS2386.LIB.
Q: How to make MAKEFILE ? I must be doing something extraordinarily stupid, but I can't help myself. My problem is the following: the make file creation creates by default a make file with the name MAKEFILE (note: no extension). This make file is used under WF without any problem. However, when I try to run NMAKE MAKEFILE standalone, it quits immediately doing nothing. Adding -a or -d doesn't change anything, except the make file name is prefixed with the "smiley face" on every message... Renaming MAKEFILE. to MAKEFILE.MAK immediately fixes the problem. What's going on?	When you set up your project, you are asked to specify the names of your target executable, etc, etc, and the name of your make file. The default for the make file is MAKEFILE (as you've discovered). Select the project in Project Control, hit the Change button, and change the name of the make file to whatever you want. The format of NMAKE format is /f makefile-filename. If you don't use /f, the default filename is MAKEFILE. WorkFrame/2 uses the /f format by default.
Q: Documentation Problem? On page 15, chapter 2 migration guide: I don't know if this is an error or not. Note I have changed the capitalization to make the strangeness visible: The operation system APIs now use the data type bitmapINFOheader2 rather than bitmapINFOheader. Solution: Change occurrences of bitmapheaderINFO to bitmapINFOheader2. Is the INFO in the right place in the solution?	Whoops! Looks like a typo to me. Thanks for pointing that out. We'll fix it.
Q: Header/Function crossref Can someone tell me where it is documented the list of which .H files are necessary for which functions? i.e., strcpy() requires STRING.H, etc. I have all the Toolkit functions covered, I'm thinking more in terms of your basic ANSI C functions.	For each function in the C Set/2 User's Guide, Migration Guide, and the SAA CPI C Reference, a prototype is given which includes the necessary header files. The prototype is right at the top of the page. There is also an appendix in both the User's and Migration Guide that lists all the functions/macros supported by C Set/2 and the header files that define each one. The Reference Summary provides this information also, i.e. the function prototype and the headers in which they are.

Question	Answer
Q: KWIKINF Whenever I start KWIKINF, I receive the error message box stating that HELPNDX was not found. I have the statement SET HELPNDX= DDE4.NDX+ EPMKWHLP.NDX in my config.sys file. The directories for all of the NDX files are included in my DPATH statement of CONFIG.SYS. Any help would be greatly appreciated.	Unfortunately, KwikINF currently has a bit of a problem dealing with multiple .NDX files. In the meantime, copy your 2 files (EPMKWHLP and DDE4.NDX) into a single .NDX file (the command is: copy dde4.ndx+ epmkwhlp.ndx newhelp.ndx) and then change the HELPNDX variable in your CONFIG.SYS file to point to newhelp.ndx, or whatever you called the new file (you can set HELPNDX on the command line too if you don't want to reboot before you use KwikINF). Make sure it's in a directory on your DPATH. Then it should work.

Q: Sending PCHAR16 as MPARAM
 	
I have the following code segment (variables changed to protect the innocent):
 	
{
PCHAR pointer32;
PCHAR16 pointer16;
MPARAM msgparm;
 	
DosAllocSharedMem (&pointer32, ... | OBJ_TILE | ...);
 	
A pointer16 = (PCHAR16) pointer32;
.
.
.
B msgparm = (MPARAM) pointer16;
 	
WinSendMsg (old 16 bit app, ..., msgparm, ...);
}
 	
When I originally ran this segment, the 16 bit app that was receiving the address on the message trapped, while a 32 bit app did not. If I print out pointer16 and pointer32 after statement A, I get the two pointer values. If I then print out msgparm after statement B, I find that msgparm actually has pointer32, not pointer16 as my assignment statement indicates. If I change statement B to:
 	
memcpy ((VOID *) &msgparm, (VOID *) &pointer16, 4);
 	
the 16 bit apps no longer trap.
 	
Everything else in the program works, I just cannot understand why the compiler does not assign the 16 bit pointer value into the message parameter variable as I had intended. Am I missing something??
	The cast to MPARAM which is void * is causing the problem. You are telling the compiler to convert the pointer back to a 0:32 pointer. Use a cast to ULONG first to stop the compiler from converting the pointer.
	
	Doing the assignment directly causes the 16-bit pointer to be converted into a 32-bit pointer. Doing a memcpy() causes the bits of the 16-bit pointer to be placed into the 32-bit pointer with no conversion. To do an assignment with no conversion, take the pointer through an integral type. Try
	
	msgparm = (MPARAM)(ULONG) pointer16;
	
	Casting from 16-bit-pointer to ULONG does not cause a conversion, and casting from ULONG to 32-bit pointer does not cause a conversion. Casting from 16-bit-pointer to 32-bit-pointer *does* cause a conversion. Doing the two casts as above allows you to move the bits directly.

Question	Answer
Q: WorkFrame/2 and C-Set/2 I have installed the Workframe product after installing the Toolkit and C-Set/2. It was installed on a LAN server, and I wanted to change the default compiler to C-Set/2. I found a profile (DDE4CSET2.PRF) on a machine that had the tools installed locally. I copied the profile to the IBMWF sub-directory created by the ADDUSER program on my LAN requester machine. I can now see C-Set/2 as a compiler option, but when I attempt to change compiler options in a project file, I get a message saying that a DLL containing compiler options can't be found. What DLL can't be found? Do I need to re-install C-Set/2 in order to make this (or other future problems) work correctly?	The DLL it's looking for is named DDE4CSET.DLL. This should be in the \IBMC\DLL directory. If not, you will have to unpack the file DDE4CSET.DL@ on the original C Set/2 diskettes.
Q: DLL problem We've installed C-Set/2 and are now migrating applications to the new environment. We ran into the following problem when creating a DLL with some functions. The produced DLL does not support the function isgraph() and toupper(). The DLL links OK but during runtime the return value from these two two functions is always 0. The initial DLL function which displayed the error is StandardiseLine(), which is in the STDLINE.C source file. If WBCisgraph() and WBCtoupper() are replaced by the isgraph() and toupper() functions it does not work correctly. It is only when the library functions are within the DLL that they appear not to work. When they are in the main module they work fine.	Sounds like the DEF file for the DLL does not specify: LIBRARY < dllname> INITINSTANCE TERMINSTANCE and DATA MULTIPLE NONSHARED

We're not ignoring you.

If you don't see your questions here, bear with us. We've a growing pile of comments, questions and suggestions for this Q&A section. We're reading and registering your ideas, and publishing them if possible.

A word from your editor

If you didn't receive this newsletter mailed direct from IBM, and you would like regular hardcopy, then let us know. Mail in the reply form at the back with your request to join us, plus your full mailing address, and we'll add you to our mailing list, wherever you are. Yes, wherever you are...from Argentina to Zaire, if you have a mail service, we'll mail you our newsletter.

For the many of you sending in the reply forms with your comments, we've often had the need to call you to discuss your comments further. It's a great help if you include your phone number. Thanks!

Next issue? C++ beta feedback, more technical articles, and lots more Q&A.....

This newsletter was produced by the OS/2 C/C++ Planning department of the IBM Canada Toronto Lab. For further information on any of the products mentioned, please contact your local IBM office, or an authorized IBM Business Partner.

This newsletter was created and marked for processing using IBM BookMaster (Program Number 5688-015) and IBM Document Composition Facility (DCF)™ (Program Number 5748-XX9).

The final copy was printed on an IBM 3825 Page Printer, an Advanced Function Printer.

OS/2 Compiler News No.3 92 November issue

Reader's Comment Form

1. *Did you find this newsletter useful?*
2. *Is there any way you think we could improve this newsletter?*
3. *Is there any compiler-related subject you would like to see addressed in this newsletter?*

Please note:

- *IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered non-confidential.*
- *Do not use this form to report compiler problems or to request copies of publications. Instead, contact your IBM representative or an authorised IBM Business Partner.*
- *If you wish, you may include your name, address, and company name if applicable, and your phone number.*

_____

_____

_____

_____

_____

Thank you for your cooperation and help. You can either mail this form to us, or hand it into an IBM office for forwarding.

You can also fax the form to us. Our fax is 416-448-6057, plus 011 44 for Canada. Please mark your fax for the attention of MJ Houghton.

OS/2 Compiler News No.3 92 November issue

Reader's Comment Form

Fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here

IBM

IBM Canada Ltd
Workstation Languages Planning
Dept 394
22/394/844/TOR
844 Don Mills Road
North York
Ontario, M3C 1V7
Canada

Fold here and tape.....fold here and tape.....fold here and tape.....fold here and tape.....fold here