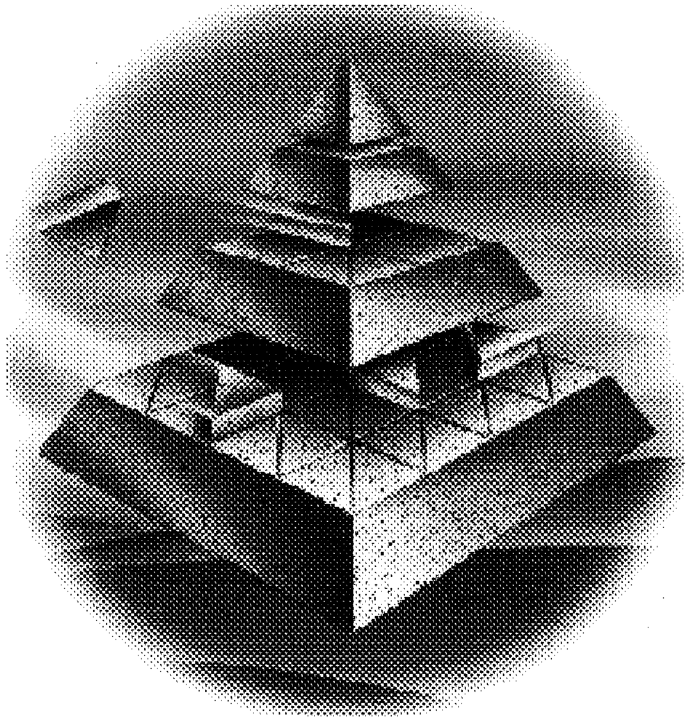

IBM C Set ++ News No.7

June 94

Weeeeee're back!

Where *have* we been, some of you are asking. It's been a quiet time on the newsletter front, and we've had a ton of mail asking if we've dried up, died, or what. It was 'or what'. We made some changes in the development area, and as a result, held off writing any more newsletters for a few months. But we're back now with a vengeance ... here's the June issue, and the summer issue is well underway. This issue, we've got :

- Debugger tips, check out page 19
- Part two of the 16 to 32-bit article that we ran so long ago, you have to go back and read part 1 again, see page 23

And ... yes, it's finally back - the Q&A section. Read on, enjoy, and keep writing to us. We read every letter. Honest.

Getting a Deal

First, let's talk about money. Saving it. There's excellent news: we're pleased to offer new ways of buying the product as well as price promotions. And the "teach yourself C++ " CD you may have heard about? That's here too, at a discount for C Set users. (See page 9 for the story of the CD.) Please be aware that not all promotions are available in all countries. If you are outside the US, please call and ask. See the article on non-US ordering on page 6 for details of where to call.

1. The C Set ++ bonus bundle

   All the C/C++ development tools you need in one package, priced very attractively. The bundle comprises the C Set ++ CD-ROM, OS/2 for

   Windows[1], the KASE:Set [2]. GUI-builder, and Experience C++ , the full multimedia C++ tutorial ... all for just ...

```
                         What to
   USA      Canada        Order       Media
   ----     ------      -------     -----
   $249     $352        83G8118     CD ROM
   $279     $395        83G8119     CD ROM+ doc
   $309     $438        83G8117     3.5"
```

   Call 1-800-3IBM-OS2 or 1-800-IBM-CALL for help in the US. This bundle is available in other countries: please call to check for availability and local prices. See page 6 for phone numbers.

2. Experience C++

   Teach yourself C++ using colour, sound, the whole works. Hundreds of screens of education, audio segments, animated films, and samples galore. Full glossary included. Existing users of C Set ++ in the US can order for $49 - call 1-800-3IBM-OS2 or 1-800-IBM-CALL for help in the US. In other countries, call to

[1] trademark of Microsoft Inc

[2] KASE:Set and KASE:VIP are trademarks of Kase Systems Inc

check for availability and local prices. See page 6 for phone numbers.

Note: This special pricing offer is not available in Europe, Middle East, Africa. Call to find out the availability of the CD and its price in those geographies.

Ask for: 03H4441 CD ROM

3. Buy now, pay less

   Promotional prices are now in effect in the US and Canada for V2.1 until December 30th, 1994.

USA	Canada	Description	What to Order	Media
$249	$353	base product	82G3732	3.5"
$189	$268	base product	82G3735	CD ROM
$219	$310	base product	82G3736	CD+ doc
$179	$254	additional lic	82G3916	n/a
$179	n/a	add lic - proof	82G3920	n/a

   Call 1-800-3IBM-OS2 or 1-800-IBM-CALL for help in the US. In other countries, call to check for availability and local prices. See page 6 for phone numbers.

4. Move up a notch

   You can now upgrade from FirstStep™ for C Set + + V2.1.

USA Price	What to Order	Media
$109	83G0008	CD
$139	83G0009	CD+ doc
$175	83G0007	3.5"

   Call 1-800-3IBM-OS2 or 1-800-IBM-CALL for help in the US. In other countries, call to check for availability and local prices. See page 6 for phone numbers.

5. Buy more, pay less

   Promotional prices also apply to the volume packs. Please note that Europe/Middle East/Africa does not sell the physical volume packs - they offer the softcopy option. See page 5 for part numbers.

USA Price	Description	What to Order	
$1145	5 pack	82G3924	
$2290	10 pack	82G3925	
$5725	25 pack	82G3926	
$11450	50 pack	82G3927	
$22900	100 pack	82G3928	
$845	5 pack	82G4451	
$1690	10 pack	82G4452	
$4225	25 pack	82G4453	
$8450	50 pack	82G4454	
$16900	100 pack	82G4455	
$995	5 pack	82G4456	
$1990	10 pack	82G4457	
$4975	25 pack	82G4458	
$9950	50 pack	82G4459	
$19900	100 pack	82G4460	
$795	5 pack	82G3929	add lic
$1590	10 pack	82G3930	add lic
$3975	25 pack	82G4443	add lic
$7950	50 pack	82G4444	add lic
$15900	100 pack	82G4445	add lic

Call 1-800-IBM-CALL for help in the US. In other countries, call to check for availability and local prices. See page 6 for phone numbers.

More from the Antipodes

Customers often remark that not everything in this newsletter applies outside the USA. I agree, and have great pleasure in announcing something that's *available only in Australia*. The big news from IBM Australia is the bundling of OS/2 for Windows and C Set + + with the crossgrade price of A$238 for the CD, and A$279 for diskette. This feature allows a current licensee of *any* C/C+ + compiler on *any* platform to upgrade across to C Set + + . This is the same price as the normal crossgrade.

And there's more ... Experience C+ + is coming to Australia - for details, see page 9 - and anyone who wants to register their interest in the CD should give their contact details to the Australian DAP at 61-2-354-7766 (fax) or 61-2-354-7684 (voice).

Rumour has it there are limited quantities of a nifty C Set + + t-shirt for the early birds who register with the DAP as C Set + + owners.

Buying in Bulk

North America

What do cereal, sugar, and C Set + + have in common? Simple. They all get cheaper, the more you buy. You won't find C Set + + on the same shelves as the groceries, (at least not yet!), but you will find it lower priced if you buy a certain number. All the various media types can be purchased in packs we call volume packs, of 5, 10, 25, 50 and 100. If you check out the price promotion in the previous article, you'll see that the volume packs offer a handsome discount over a single copy. You get the same product as the single copy, just more of them. And they are single copies in individual boxes - if you order a five volume pack, you get 5 singles in their own boxes.

Volume packs are available through 1-800-IBM-CALL in the USA, 1-800-465-7999 ext 690 in Canada, or your IBM Representative.

Outside North America

In Europe, Middle East and Africa, the physical volume pack is not available. By physical we mean that you receive the actual package when you order. What is available is the softcopy option - you can copy the product with IBM's permission. You buy the right to make one or more copies, and IBM doesn't actually ship you the product.

The softcopy option is available in a volume pack.

Part number	Pack size
72G6896	5 licences
72G6897	10 licences
72G6898	20 licences
94G1633	50 licences
94G1634	100 licences

See page 6 for where to call for help in ordering and prices.

North of the Border

Usually when we quote prices, we use the US dollar, and remind you that the quotes are for US consumption only, and that prices elsewhere in the world are different. When it comes to Canada, the US dollar price doesn't apply either. We share a continent, a border, and a love of baseball and hockey, but we don't share a currency or pricing. Canada has its own pricing for C Set + + . As you've read earlier, C Set + + V2.1 can be obtained at promotional prices until the end of 1994, but did you know that FirstStep is also available at prices lower than those originally announced, and these new prices are in place until further notice? Here's the scoop on what it costs to buy FirstStep in Canada using Canadian dollars:

```
              C Set + +   FirstStep

Part number   What it is          Old price     New Price
-----------   ----------          ---------     ---------
  82G3744     3.5"                Cdn $193      Cdn $150
  82G3746     CD ROM              Cdn $167      Cdn $110
  82G3747     CD ROM+ doc         Cdn $193      Cdn $150
```

This is only a short list of Canadian prices - the volume packs, additional licences, and other items are also available at new prices. Check IBM Direct at 1-800-465-7999, your IBM Representative, or your dealer, for details.

Get it from the Source

Yes, Virginia, you can get the source code to the User Interface and Collection Classes Class Libraries. Ask for part number 61G1400. Who do you ask? In the USA, call 1-800-IBM-CALL; in Canada 1-800-465-7999. In other countries, check your local IBM Direct. Some countries in Europe may be able to order direct from IBM Copenhagen. And of course, your dealer has it.

Non-US phone numbers

For those of you not in the US, there are a host of phone numbers you can use to order C Set + + products.

Country	Phone Number
Austria	0222 21145 2500
Australia	008-805-109
Argentina	319-6480
Belgium	02-225-33-33
Canada	1-800-465-7999 x690
Denmark	8030-4545
Finland	90-459-6900
France	05-03-03-03
Germany	0130-812177
Greece	30-1-328-1342

Country	Phone Number
Hong Kong	852 825-6268
India	91-80-526-7117
Indonesia	62-21-520-9500 x1360
Ireland	353-16603744
Italy	1670-17001
Japan	0120-04-1992
Korea	82-2-781-6555
Malaysia	03-717-7788
Mexico	627-1028
Netherlands	030-384040
Norway	66-99-93-00
Philippines	815-4119
South Africa	011-224-2000
Spain	900-100-400
Switzerland	01-436-6233
Sweden	08-793-4004
Taiwan	886-2-776-7878
Thailand	66-2-273-4444
UK	081-575-7700

Doucher Down Under

We C Set ++ folks certainly get around. And not just in North America. Your editor was in Europe a few months back, along with a senior technical person, showing off C Set ++, meeting with customers, and helping local IBM staff understand the product.

More recently, Tom Doucher, the overall manager of C Set ++ for OS/2 also hit the road, this time to Australia. Tom is the Product Manager for C Set ++, which means that he owns the development plan for the product. *That* means he says what gets done .... definitely the man to talk to if you see him! Tom's travelling companion was one of the key people on the C compiler front end - Dave Mooney. Many users know Dave from his appearances at shows; many more know his name - and his style - from the IBMLink™ C-SET2 forum. This time it was the turn of the "land down under" to meet Dave.

Tom & Dave's main focus was to kick off the availability of V2.1 in Australia, and they had a week to do this in. Focusing on Sydney & Melbourne, they crammed their week with everything they could set up - gathering customer requirements, giving demos, attending a User Group meeting, and holding a press review,

Tom paid special attention to the key issue of availability of the product down under, meeting with both IBMers and customers to review the concerns. These included:

- a long delay between announce and availability
- perceived high prices
- why it seems faster for a customer to buy in the US than locally in Australia
- availability of local support
- access to corrective service

On his return, Tom reviewed his findings with the C Set + + team, and work is underway to address all the concerns.

The visit showed that our antipodean friends have taken C Set + + to their hearts - Tom and Dave could have easily stayed a lot longer. Perhaps next time.

Alistair and Alex in Asia

Alistair Rennie, the Planning manager for C Set + + was also around and about recently. He and Alex Wong (of our C Set + + Build & Test department) had the opportunity to visit Taiwan, Korea and Indonesia to present the complete C Set + + story to the IBM technical and marketing teams, the local dealers and customers.

OS/2 is really getting underway now in those countries, so there's a great deal of interest in C+ + and the product strategy, as well as competitive advantages and futures.

When C Set + + people are in town, it's an opportunity to talk about requirements, and on this trip, DBCS was a key focus item. Other issues came up too:

- The provision of Corrective Service.
- Inventory, and leadtimes to obtain the product
- Upcoming betas
- The bundles we mention in this newsletter
- DAP programs

Alistair and Alex came back with lots of to-do's and we'll keep you posted of any developments.

Experience C++

What it is

Experience C++ is a Multimedia CD-ROM that teaches C++ programming. It does assume some programming knowledge but does not assume a knowledge of C. Many sample programs are used to teach concepts. All sample programs are included in complete source form as well as executable. While in the package, the user can have the source explained, view the actual source file and run the program.

Three C++ compilers are supported

1. IBM C Set ++ for OS/2 Release 2.0
2. Borland Turbo C++ for DOS 3.0 Release 3.0
3. Microsoft Visual C++ Professional Edition Release 1.0

The CD has

- 18 chapters
- 369 screens of education + 20 menus
- over 1400 audio segments
  - 177 pages of text in closed caption
  - 6.5 hours of audio
- 23 animation films
- over 90 sample programs in source and executable form
- glossary with over 320 definitions
- hotlinks to move around and an index to find topics

What hardware you need

The minimum requirements for running EXPERIENCE C++ are any IBM PS/1 or IBM PS/2 model, or 100% compatible, with the following minimum configuration:

- Intel386 SX - compatible or higher based personal computer
- 2MB memory
- 500K of program memory available
- 800K of available extended or expanded RAM. Use the DOS command MEM to list the memory characteristics of your machine.
- IBM VGA display adapter, or compatible, color or mono

- A DOS or OS/2 compatible CD-ROM drive

Optional items

- IBM Mouse, or compatible
- One of these audio cards
  - Digispeech DS-201A
  - IBM Audio Capture/Playback Adapter
  - IBM Audio Capture/Playback Adapter/A
  - IBM PS/1 Audio/Joystick Adapter
  - SoundBlaster 1.0, 2.0, or compatible
  - SoundBlaster PRO (also Pro OPL/3)

What software you need

The minimum requirements are:

- DOS 5.0 (or higher) installed
  or
- OS/2 2.0 (or higher) installed
- Sufficient DOS environment space to create one DOS environment variable (only required for caption-only mode)

Support

Experience C++ service questions should be directed to:

Mark Ryan	John Akerley
(416) 448-3378	(416) 448-3757
mryan@vnet.ibm.com	akerley@vnet.ibm.com

The questions that users ask about Experience C++ are almost always related to sound. Most users run into problems because they either specify the wrong sound card during installation, or do not specify the sound card options correctly. The READ.ME file on the CD discusses these problems in detail. Here is a list of the most common problems:

Problem	**Reason/Details**
E 04 Error at startup	Not enough base memory available in DOS. The READ.ME file on the CD has complete details. Use MEM command to determine how much current base memory is available. If less than 500K, you can suggest the following to make more memory available: 1. MEMMAKER (for MS-DOS) or RAMB-OOST (for PC DOS) maximize the base memory that is available. 2. Comment out DEVICE= statements in CONFIG.SYS for drivers that you do not need. 3. Load drivers high where possible.
E 03 Error at startup	Experience C++ can't communicate with the sound card because the audio support was not installed or was installed incorrectly. Either the audio support has not been installed (in which case you can see the inside cover of the CD for installation instructions), or the installation did not work properly. The most common installation problems are improperly specified IRQ, DMA, and I/O addresses. The READ.ME file on the CD contains a section called "Experience C++ Error Codes" that describes these problems, and how to fix them, in detail.
Won't start/locks in Windows	Conflict between Experience C++ and other Windows applications that use the sound card. Experience C++ needs the sound card all to itself to run under Windows. If you are using another Windows application that needs the sound card, it is best to start Experience C++ directly from DOS. More details are available in the "Problems Running Experience C++ in Windows" section of the READ.ME file on the CD.
Scratchy sound	The SoundBlaster compatible sound driver works with a wide variety of sound cards, but the sound quality may be poor on some systems. You can get Experience C++ to use an improved sound driver by specifying "SoundBlaster 16 and 100% compatible" in the DOS and OS/2 install procedures. However, this sound driver only works with genuine SoundBlaster 16 cards and cards that are 100% compatible.

Problem	Reason/Details
Can't specify IRQ = 10 in installation procedure	The sound drivers that Experience C++ uses cannot accept an IRQ higher than 7. If your sound card is set to IRQ 10, the sound drivers cannot find it, and you will get an error if you try to start Experience C++ in audio mode. You can still use the tutorial in closed-caption mode, or you can try resetting the IRQ on your sound card to 5 or 7 using the SBCONFIG command. Note that this may cause IRQ conflicts if both of these IRQs are already being used.
Experience C++ hangs system	Sound card specified during installation is not the sound card that the system actually uses. Experience C++ can't detect if the user specifies an incorrect sound card during the installation procedure. If an incorrect sound card is specified, Experience C++ may hang when the user tries to start it. Reinstalling with the correct sound card specified should correct the problem.

Where are those EMEA DAP folks?

Looking to talk to the people who run the EMEA DAP? There's a generic fax inquiry number for them - 44 (UK) 256-336778. However, you might want to try their fully automatic voice response unit at 44 (UK) 256-50096. It's a helpfax that guides you through various options and can send you forms and other information, a 24 hour a day, 7 days a week service. Isn't technology wonderful?

HiTest Tools

Edge Research Inc. has released HiTest Tools, a set of development tools for Lotus Notes. The core product in the suite, the HiTest API, supports the C Set ++ compiler, and Notes programs built with that compiler run over three times faster than the same programs built with other compilers.

The HiTest API, layered on top of the regular Notes API, has error handling, an object model and support for Microsoft and IBM compilers. However the Windows and OS/2 1.3 versions are 16 bit implementations, while the OS/2 2.1 version supports 32 bit programming when used with the C Set ++ compiler. There is single function support for operations such as macro execution, full text search, sending mail and composite import/export. There are also functions for metadata browsing, data access with automatic data conversion and for high level manipulation of rich text. API programs performing complex tasks such as composite import or macro execution take 15 or 20 pages using the Lotus V3 API but only 15 or 20 lines using HiTest, and the code flows much more smoothly.

The HiTest Tools can be downloaded from CompuServe, Ziffnet, anonymous FTP on the Internet, and from the Edge Research's Notes databases on WorldCom. There is no licensing charge for the first two copies at any one company. On CompuServe, look in the Lotus Communications+ Forum, in the Notes API/Dev library section. On the Internet, go to the pub\edge directory on Worldcom.com.

Edge's WorldCom distribution database, the best place to look, has a Product Warehouse view from which products can be detached, a Release Notes view, a Sample Applications view, a Discussion view, and a Support bug report/response view. Companies that choose to deploy the tools suite beyond the first two copies can purchase licenses in bulk, starting at 100 seats. Introductory pricing through August 31, 1994 for 100 seats is $1995 US.

Edge Research is a privately held company that was founded in 1993 to build Notes tools and utilities. In addition to its product development activities, the company conducts a substantial Notes-oriented consulting practice. Edge runs an electronic enterprise, distributing its software only through computer download. Its offices are at One Harbour Place, Suite 455, Portsmouth NH 03801. Edge can be reached at 603-431-5300 (voice) or 603-427-2541 (fax).

Decisions, Decisions - Customers Rule OK!

If you're a reader of this newsletter from way back, you'll know that time never stands still here in C Set + + -land. No sooner do we have one release out the door, then we're thinking about the next one. Now, don't get excited; this article isn't to tell you what's coming down the pipe, at least not in detail.

What we're looking for is your input on some changes we *might* make to existing features. We know that as you've spent time and effort in coding millions of lines of code, you'd rather we didn't introduce something that sends you back to the drawing board!

We'd like to hear from any of our customers who are using the _parmdword built-in function, or the task standard class library (aka the AT&T or USL[3] task class library).

_parmdword Built-in Function

The _parmdword built-in function is used to access the number of dword parameters passed to a _System linkage function. This feature was intended as an aid to programmers extending APIs, whilst maintaining backwards compatibility. The size of the parameter list could be used to determine which version of the function was being called.

To implement this function, the size of the parameter list passed to a _System linkage function is always passed to the function in the AL register. This has obvious performance costs for all _System functions, both in terms of slower execution speed, and larger code size. Also, we're not aware of many customers using this feature. Thus, we're thinking of making _parmdword support optional. The support would be enabled or disabled by specifying an option. Note that if a _System linkage call is compiled with _parmdword support off, and the called function uses the _parmdword function, the return value from the function will be undefined.

Do you use _parmdword support? What are you doing with it? Would you have a problem if we put the support under an option? Should the function be enabled by default? (this is what we're thinking of doing).

AT&T Task Class Library

The task library was designed as a portable implementation of single process multitasking. Since thread support in OS/2 is a powerful, native multitasking mechanism, we suspect that few of you are using the task library for OS/2-only applications. Similarly, many other platforms also have their own multitasking support, which you may prefer to use instead. So, it isn't clear to us whether the task library is important to our customers.

Because the task library is quite expensive to port, we'd like to be sure it is a vital part of the product before we begin moving it to other platforms.

Please tell us whether you are using the task library. What are you using it for? Do you use it on multiple platforms? If so, which ones?

[3] USL is a registered trademark of UNIX Syxtems Laboratories

If we removed the task library from C Set + + , would this cause you a serious problem?

If the above two functions mean a lot to you, please let us know. Send your two cents/pfennig/lira/ore etc worth to:

- Internet[4] - slauzon@ibm.vnet.com
- CompuServe[5] - 73251,1733 ; this is Roger Pett's id, so please mark any mail for the attention of Stephen Lauzon
- IBMMail - CAIBMS2M @ IBMMAIL
- Fax - (416) 448-6057 Attn: Stephen Lauzon

And you can always write using the reply form at the back of this newsletter.

Introducing ...

...Custom Application Porting Workshops for OS/2™

IBM and One Up Corporation are cooperating in an effort to provide porting assistance to ISVs and corporate accounts. IBM's new individualized workshops simplify the process of porting your existing applications to OS/2 32-bit, with native source code, from other platforms.

IBM's technical experts will assist you in creating 32-bit OS/2 applications based on existing 16-bit OS/2, DOS, Windows 3.x, or UNIX applications. In addition, technical teams can help developers to use OS/2's System Object Model (SOM)™ and Pen computing capabilities.

Porting efforts to OS/2 as a result of the relationship of IBM and One Up over the past two years have achieved 100% customer satisfaction.

The new individualized services replace classroom-style group workshops IBM has offered in the past as part of its Developer Assistance Program.

Custom Application Porting Workshops for IBM OS/2 will:

- Accurately size your project
- Minimize the learning curve

[4] Registered trademark of Internet

[5] Trademark or registered trademark of CompuServe Inc

- Guarantee architectural integrity
- Shorten development schedules
- Port anywhere from 35% to 100% of your application during the workshop
- Introduce you to the latest porting tools and techniques

The five phases of the new custom porting

- Phase 1 - Analysis

  Analysis of code to identify and report specific issues and identify amount of porting effort required. This includes a breakdown of all API calls, type definitions, symbols and messages. The analysis provides a detailed look at the source, providing a printed report. This phase uses "S.M.A.R.T. Ally".

- Phase 2 - Automated Code Replacement

  This phase includes automated code replacement of those items that have a one-to-one mapping from the source to the target environment. (The average has been 35% to 65% of the code.) Also included in this phase is the conversion of resource files. This phase uses "S.M.A.R.T." Port.

- Phase 3 - Computer Assisted Code Replacement

  This phase includes interactive code replacement, with input from an application developer for those source items that have an equivalent feature in the target environment, yet require a decision.

- Phase 4 - Implementation of Unsupported Features

- Phase 5 - Addition of Platform Specific Features

Finding out more - USA

For more information and to register, call One Up Corporation's toll-free number, 1-800-678-31UP, and refer to the Custom Application Porting Workshops for IBM OS/2 Operating System.

Finding out more - Worldwide

Outside the U.S., please call 214-620-1123, extension 2500. Or contact the workshop team directly: Marilyn Johnson, Program Manager, (407) 982-5514, Laura Rose (407) 443-1640, Terry Kemmerer (407) 982-1041. If you prefer electronic, you can reach the team at mjohnson@vnet.ibm.com.

OS/2 & LAN Systems Development Tools Guide

The OS/2 & LAN Systems Development Tools Guide is a treasure trove of tools for application developers. It tells you about the many tools, produced by independent software vendors and IBM, that can help developers reap the benefits of the advanced technologies in OS/2 2.1, LAN Server 3.0, and related products.

The Guide features detailed information about more than 400 OS/2 and LAN Systems tools and utilities divided into 35 categories and indexed by both company and product name.

The Guide comes both in hardcopy and in a softcopy OS/2 .INF file.

Hardcopy

In the USA: You can order the guide by calling either 1-800-444-4881 (Miller Freeman Inc.) or 1-800-879-2755 (IBM Software Manufacturing Solutions). When calling the IBM number, specify publication G362-0025. There is a 9.95 US$ charge (plus shipping, handling, and applicable sales taxes).

Outside the USA: Customers outside the USA may order publication G362-0025 from the IBM Software Manufacturing Solutions in Copenhagen, Denmark.

Additionally, copies may be obtainable from some of the Developer Assistance Programs in other countries and geographic areas. Contact them for information. See page 12 for details on how to contact the EMEA DAP.

Softcopy

The OS/2 .INF softcopy version of the Guide is not available from the above mentioned places. It will be distributed at no charge on various e-mail and BBS systems, wherever the monthly electronic PSP Developer Support News (DSNEWS) is distributed. The file name is OS2TG.

Contents of the ZIPped File OS2TG.ZIP

This announcement file is part of a ZIPped package that also contains the Tools Guide in .INF format and several formats of the product nomination form. When UNZIPped, the files are:

Filename	Size	Description
OS2TG.ANN	3665 bytes	This announcement file
TG.INF	539749	The Tools Guide in .INF format (To look at this file from within OS/2, type VIEW TG)
NOMINATE.TXT	10656	Nomination form in ASCII format
NOMINATE.EXE	186877	Self-extracting nomination form in Encapsulated PostScript format (executing NOMINATE produces the .EPS file)

Note: Within the file TG.INF is a version of the product nomination form that you can print using the OS/2 VIEW facility.

If you are unable to print and use any of the formats of the product nomination form included in this ZIPped file, we will be glad to send you a fax copy of the form. Call the US IBM Developer Assistance Program at (407) 982-6408.

Call for Tools and Applications to List in Next Issues

IBM Software Developer Support is currently soliciting tools developers to list their products in the next edition of the OS/2 & LAN Systems Development Tools Guide.

We are also accepting product nominations for listing in the OS/2 Applications Directory. This book includes development tools as well as 32-bit OS/2 and LAN Systems applications.

To nominate your application for listing in the OS/2 Applications Directory, or your development tool for listing in both guides, use the product nomination form files listed above.

The IBM contact for this guide is Mike Engelberg. Mike can be reached at (407) 982-0500 (voice) or (407) 443-4233 (fax), as well as

- Internet - dsnews@vnet.ibm.com
- CompuServe - 74150,44
- BIX - dsnews@bix.com
- IBMMAIL - USIB33NP

CICS OS/2 V2 C++ Support

Are you a CICS OS/2 user? Have you been asking for OO support? You're in luck. Since March 31, 1994, support for the C++ language has been available, in direct response to customer demand for CICS application support using OO methods. These methods, using class libraries to encapsulate function, allow greater reuse of code.

CICS transactions, running on CICS OS/2 Multi-user and CICS OS/2 Single-user servers can now be written in the C++ language. This means that OO programs can now access the CICS API. You get the increased performance and flexibility of 32-bit CICS applications, plus the benefits of OO development methods.

In addition to C++ support, there's also a new user exit (exit 27) available which enables multi-session debugging, very helpful for debugging client/server applications which use the External Call Interface (ECI) or the External Presentation Interface (EPI) for communication between the front-end application on the client and the CICS application on the server.

The C++ support and multi-session debug are supplied on corrective service diskettes. US customers can obtain these by calling the IBM Support Centre at 1-800-992-4777. You will be asked for your name and number, product name and number, and the CSD number you want. That's UN59266 for single-users, and UN59267 for multi-users. Outside the US, customers should check with their normal support service.

The CSD also includes softcopy publications on these new features, and these pubs will also be available in the future on the Transaction Processing and Data Collection Kit CD ROM.

Do the Debug

We're going to review some hints and tips on the debugger, but before we do, some answers to questions that pop up all the time:[6]

- IPMD is the same as the C Set ++ debugger is the same as the PM debugger. They're just different names for the same thing. IPMD stands for Interactive PM Debugger. End of mystery!
- IPMD works on C Set/2 code as well as C Set ++.
- IPMD can debug 16 bit code containing Microsoft Codeview format debug information.
- IPMD is certainly part of C Set ++. The reason why you may hear the location Lexington, Kentucky, mentioned along with IPMD, is that the folks on the C Set ++ team who developed the debugger are based in that town.

Now, on with the hints.

[6] The tips in this article have also been published in the DevCon newsletter.

Child processes

DosExecPgm allows your program to request that another program execute as a child process. With a simple change to the parent program, you can start an instance of IPMD to independently debug the child process. The following code segment demonstrates the use of DosExecPgm to start the program "t.exe":

```
int i;
char LoadError. 100";
RESULTCODES rcodes;

i =  DosExecPgm(LoadError,
               sizeof(LoadError),
               EXEC_SYNC,
               NULL,
               NULL,
               &rcodes,
               "t.exe");
```

A simple change to 2 lines of the DosExecPgm call produces the following:

```
i =  DosExecPgm(LoadError,
               sizeof(LoadError),
               EXEC_SYNC,
               "ipmd.exe\0-n t.exe\0\0",
               NULL,
               &rcodes,
               "ipmd.exe");
```

The "-n" is one of the optional command line parameters to IPMD which indicates that restart information is not to be used.

IPMD does not have to be active when this code is executed but, if it is, a second complete instance of IPMD will start executing. The only restriction in this comes from DosExecPgm - you may not use DosExecPgm to start a process that is of a different type than the starting (parent) process. Since IPMD (the child) is a PM program, the parent must also be PM. This is easily accomplished by using the "/PM:PM" switch when linking the parent.

Typecasting

The Monitor Expression dialog allows you to enter expressions for evaluation and monitoring. If you need the address of variable "parray", you enter "&parray" in the Monitor Expression dialog and the address is displayed in a monitor window. The expression evaluator supports a subset of the C/C++ language.

One of the major functional enhancements in the C++ release of IPMD is the ability to perform user typecasting in the Monitor Expression dialog. In order to use complex typecasting your application must be compiled using the C+ + front end (rather than the C front end). This is, of course, automatic for C+ + programs and can be requested for C programs with the /Tdp compiler switch.

A very common use for such typecasting is to provide for the display of dynamically allocated arrays. Consider the following program:

```
#include < malloc.h>

int main(int argc, char **argv,char **envp)
{
   int i =  500;
   char * carray =  (char *) malloc(i * sizeof(char));
   carray< 0>  =  'a';
   carray< 25>  =  'z';
}
```

The standard method for displaying a variable with IPMD is to place the mouse pointer on the variable name and double click the left button. If such an action is performed on the variable "carray", the program monitor window displays only the first element of "carray" (as shown on the top line of the Program Monitor window in the first example.

You can look at individual elements by typing "carray< 10> ", for example, in the Monitor expression dialog or you can determine the address of the array and use the storage window to view the contents.

There is an easy way, however, to display the entire array in the monitor window. If the following expression is entered in the Monitor Expression dialog, any number of the array elements will be displayed in the Program Monitor window (after the representation is changed to "Array"):

```
      *(char(*)< n> )carray
```

Where "n" is a number representing the number of elements you want to see.

Debugging PM Applications

When debugging a PM application, IPMD functions precisely as it would in debugging any OS/2 application, and no additional preparation is necessary on the part of the user. The debugger automatically detects that the application is a PM application. However, there are a few additional concepts of which you should be aware, and these are described below.

Presentation Manager message-based system. As program events are encountered by PM programs, the programs communicate with each other by passing messages and by receiving user input through input messages. When a PM program encounters an enabled breakpoint, the input queue can become blocked and dependent program events, or processes, can also become blocked as a result. For example, the input queue can become blocked when your program is halted at a breakpoint that has been triggered by an input event. The PM debugging option of IPMD is built around two distinct ways of managing these dependencies.

IPMD provides two modes of operation - synchronous mode and asynchronous mode - which refer to different means by which synchronous message dependencies are handled when the debugger has control (i.e. the application is at a breakpoint, stepping, etc).

In synchronous mode, the synchronous dependencies are held for all applications except the debugger. More precisely, the keyboard and mouse messages for the debugger are processed but those for other applications are not. Thus all other PM applications, though still processing generally, will freeze when a synchronous dependency is hit. In particular these applications will not respond to mouse and keyboard messages as long as the debugger has control.

When the debugger has control in asynchronous mode, it responds to all messages on behalf of the application in a default fashion. Thus all other applications will behave normally. However, because the program being debugged is suspended and because the debugger is providing a default response to messages, the user cannot effectively operate on the application's windows while the debugger has control. If you want to look at the window your application is executing in, uncovering it by moving other windows won't help. Since no repaint messages are being processed, you are left with a copy of the moved windows on top of your application's window! To view the application's window when the application is stopped, position it so that it is not covered when IPMD gets control.

Do not operate the debugger in asynchronous mode if the PM application that you are debugging requires the appropriate response to its messages. For example, a dynamic data exchange (DDE) message would require the appropriate response.

The PM Debugging mode selection dialog is accessed from the Session settings menu under the Options pull-down.

REXX DLLS

You can easily debug DLLs called by REXX programs. Start IPMD with the following syntax:

```
IPMD CMD.EXE /K < your REXX.cmd>
```

When IPMD displays the disassembly code for CMD.EXE, set a load type breakpoint to stop on the load of the DLL you want to debug. Tell IPMD to Run. At the popup indicating your DLL has loaded, set any necessary breakpoints in the DLL's functions.

Double the bits, double the fun? (16 going on 32, Part II)

At long last, the second part of our series on 32-bit and 16-bit mixed-mode programming has arrived. In part 1 of the article (you can find that in Newsletter number 3), we covered the basics of declaring and calling 16-bit functions. In this article, we'll look at it from the other end, and discuss calling into 32-bit code from 16-bit.

This unique feature of C Set ++ allows 32-bit applications to use 16-bit subsystems which employ user callbacks. In other words, you can call a 16-bit subsystem, and pass it the address of one of your application's 32-bit entry points, which the 16-bit code can call back to when necessary. This support is also useful when migrating a 16-bit application to 32-bit code. You can convert portions of your 16-bit application to 32-bit, and call between the 16-bit and 32-bit portions.

Now for the details:

What you can do in 16-bit callback functions

- declare a 32-bit function as a 16-bit entry point
- call these 16-bit callback functions from 16-bit code

What you can't do

Callback functions can in general accept any type of parameter and return any type. However, there are a few small restrictions:

- structures or unions cannot be passed as parameters (you can however pass a pointer to these aggregate types)
- structures or unions cannot be returned from callback functions (again, pointers to these types can be returned)
- no more than 120 bytes of parameters (note that's 30 long integer parameters!)

- C+ + member functions cannot be given 16-bit linkage

What types of 16-bit callback functions are available?

Two linkage conventions are supported by C Set + + for 16-bit callback functions:

- 16-bit C linkage - the standard C convention, parameters pushed in right to left lexical order, caller cleans up the parameters
- 16-bit Pascal linkage - parameters pushed in left to right lexical order, callee function cleans up the parameters

How do you declare a 16-bit callback function?

Remember the 16-bit linkage keywords you use to declare a function as function? Well, to declare a 32-bit function definition as a 16-bit entry you simply use the 16-bit linkage keywords in the function definition (for the newer C programmers out there, a function definition is a declaration of a function that actually contains the body, or code of the function).

In case you've forgotten, the 16-bit linkage keywords are:

Keywords	Linkage
_Far16 _Cdecl	16-bit C linkage
_Far16 _Pascal	16-bit Pascal linkage

So, to define ***daniel*** as a 16-bit Pascal callback function taking a long integer parameter and returning a long integer, you would use:

```
long _Far16 _Pascal daniel(long  cheryl)
{
   return(cheryl);
}
```

These functions will of course be called from 16-bit code. Thus, any pointers passed to a 16-bit callback function will be stored as segmented (16-bit segment selector, and 16-bit offset) pointers, not as the flat (32-bit offset) pointers used in 32-bit code. Similarly, any pointers returned by 16-bit callback functions must be segmented.

To indicate to the compiler that the incoming and outgoing pointers are segmented, use the _Seg16 keyword on the parameter declarations, and function definition. Thus, to declare a _Cdecl callback that has a parameter that is a pointer to a structure, and that returns a pointer to an integer:

```
typedef struct st {int s1;} struct_type;
int i;
int * _Seg16 _Far16 _Cdecl andrew(struct_type * _Seg16 hester)
{
  i = hester-> s1;
  return(&i);
}
```

Be careful, as with all C Set + + type qualifiers, _Seg16 binds to the left, so the _Seg16 must come after the * being modified (try putting it before the * and see what happens!).

When you use a _Seg16 pointer in 32-bit code, C Set + + will automatically convert the segmented pointer to a flat pointer that can be used in 32-bit code. Did you notice that when we dereferenced the **hester** pointer in our previous code snippet, we did *not* have to cast it to a non-_Seg16 pointer? Similarly, the compiler will convert flat pointers to segmented when their value is assigned to a _Seg16 pointer.

How do you declare and call a 16-bit callback function from 16-bit code

To a 16-bit compiler, a 16-bit callback function is just another run of the mill 16-bit function. Thus, in 16-bit code you declare a 16-bit callback function with the same keywords you would use in declaring a 16-bit function. For example, to declare the **andrew** function discussed earlier in Microsoft C 6.0, or IBM C/2:

```
typedef struct st {int s1;} struct_type;
int _cdecl *andrew(struct_type *);
```

and to call it:

```
int j;
struct_type ss;
ss.s1 = 4;
j = andrew(&ss);
```

Using a 16-bit callback function as .... a callback

As you can probably guess from the name, this is the most common usage of the 16-bit callback functions. As with any callback scenario, the process is to pass the address of the callback function into the subsystem (in this case, a 16-bit subsystem), which will store the address in a function pointer. When the subsystem needs to, it can then dereference the function pointer to call back into your application. In case you didn't follow that, here's a simple example. First, let's look at the code on the 32-bit side:

```
#include < stdio.h>
void _Far16 _Cdecl RegisterCallback(void (* _Far16 _Cdecl)(long));
void _Far16 _Cdecl Call16BitSubsystem(void);
void _Far16 _Cdecl CallbackFrom16Bit(long lParm)
{
   printf("back in 32-bit - here's our parameter %d\n", lParm);
}

main()
{

 RegisterCallback(CallbackFrom16Bit);
 Call16BitSubsystem();

}
```

The first two function declarations declare 16-bit functions. Remember that since these statements are function ***declarations*** they refer to external functions in 16-bit code, not 16-bit callback functions. ***RegisterCallback*** takes a pointer to a void Cdecl function with a long integer parameter. We'll use this function to store the address of the 16-bit callback function. ***Call16BitSubsystem*** is the entry into our (very small) subsystem.

CallbackFrom16Bit is our 16-bit callback function. As you can see, we used the 16-bit linkage keywords on the function ***definition*** to make this a 16-bit entry in 32-bit code. Finally, the main program registers the callback, and then calls the 16-bit subsystem. Don't forget that the use of a function name without parentheses (the function call operator) is equivalent to taking the address of the function.

Now, on to the 16-bit code itself:

```
void (long _cdecl *CallbackToCall)(long);
void _cdecl RegisterCallback
             (void (long  _cdecl *PointerToCallback)(long))
{

   CallbackToCall =  PointerToCallback;

}

void _cdecl Call16bitSubsystem(void);
{
    CallbackToCall(1);
}
```

Since this is 16-bit code that will be compiled with a 16-bit compiler, the syntax is slightly different from the normal C Set ++ syntax. This code is written for MS C 6.0, or the old (deceased) IBM C/2 1.1 product. The ***_cdecl*** keyword is the equivalent of the _Far16 _Cdecl keyword in C Set ++. If you look carefully at the declarations of the function pointers, PointerToCallback, and CallbackToCall, you'll notice that in these compilers, the linkage keywords must be placed to the left of the pointer being modified.

As we mentioned earlier, the ***RegisterCallback*** function stores the address of the callback function in the CallbackToCall pointer. ***Call16bitSubsystem*** then calls back into 32-bit code using the pointer.

What type of 16-bit code can call 32-bit code?

You can call from any type of 16-bit code into 32-bit code. The 16-bit code can be in an executable or a DLL. The entry point of the application can be 32-bit, or 16-bit. However, do remember that due to the restrictions discussed in Part I, the 16-bit portion of your application, and the 32-bit portion should be dynamically linked.

And yes, you can call a 16-bit callback functions from 32-bit code, both directly, and indirectly. Just declare the functions as you would a 16-bit function.

Wait .... there's more!

Until recently, 16-bit callback functions that were located more than 64KB from the start of your executable's code segment were not called correctly. But as of CSD CTC0008 of C Set ++ V2.x (we're up to CTC0009 now), this problem has been fixed, and your callback functions can now be safely located anywhere.

Well, that's all for now. We'll call back to you in an upcoming issue, with a description of sharing data between 32-bit and 16-bit code.

Now Hear This

Our users are always coming up with something others can make use of, and sometimes they ask us if we'll help get the word out. If it meets our criteria for publication, we're happy to publish the hint or pointer. This is one of those occasions.

The TMap Utility

TMap is a small utility program that uses the OS/2 container control to display some of the information from the DB2™ system catalog tables. It creates a tree view of userid's, each of which can be expanded to show the tables they created. Selecting a table creates a details view of some of the attributes of its columns. While it is useful in its own right, TMap is intended as a small example program demonstrating the use of C Set ++ and the UICL to retrieve and display information from DB2/2™ tables. It is companion to an article on the subject which is expected to appear in the July/August issue of the OS/2 Developer magazine, and is offered as such on an as is basis; all customary caveats apply. The source can be obtained in machine readable form from the OS/2 Developer forum on CompuServe (OS2DF2), the OS2BBS on IBMLink, or directly from the author over the Internet.

The Author

Liston Tatum is a computer systems senior engineer with the Information Systems Division of the University of Virginia's Health Sciences Center, which is a large teaching hospital and regional tertiary care facility located in Charlottesville, VA. He can be reached on the Internet as glt@dmt03.mcc.virginia.edu, or via IBMMail as USUVASHB.

Colorado Springs Eternal

The third annual OS/2 developers conference known as ColoradOS/2[7] is creeping up on us. The date has been set for the week of October 30, through November 4, 1994, at the Cheyenne Mountain Conference Resort. The C Set ++ team will have representation, and we look forward to seeing you there and talking C Set ++ all day long.

There's more than C Set ++, though. For starters, the keynote speakers:

- Grady Booch, Chief Scientist at Rational
- Mike Cowlishaw, an IBM Fellow, the creator of REXX
- Lois Dimpfel, Director of Personal Operating Systems, PSP
- Jed Harris, Executive Director of Component Integration Lab
- Cliff Reeves, Director of Object Technology at IBM
- John Soyring, IBM's Director of Strategic Relations in PSP

Then, there are over 40 presenters covering 80 topics, everything you wanted to know about OS/2 and were afraid to ask. C Set ++ will be covered in several sessions, and specialists in the User Interface Class Library will be there to teach and talk. Non-C Set ++ topics include:

- OpenDoc [8]
- SOM/DSOM
- Workplace OS
- Internals of HPFS
- Client/server applications
- Device drivers for OS/2 and WPOS

[7] ColoradOS/2 is a trademark of Kovsky Conference Productions

[8] OpenDoc is a trademark of Apple Computer Corp

- Smalltalk [9] and OS/2
- GPI programming
- Workplace Shell programming
- Object REXX, and integrating REXX with OS/2 applications
- OS/2 and printing

And much more. Excited? The brochure with final details (things do sometimes change) will be out in July. Fax your name and address to the USA, 719-481-8069, and one will be sent to you. If you have other questions or need to talk with the organiser, Wayne Kovsky, call 1-800-481-3389 in the USA and Canada, 719-481-3389 elsewhere in the world ... which reminds me, a full quarter of attendees last year were from outside North America, so if you're thinking of coming from outside the USA and Canada, don't worry, you won't be on your own! Last year, the Australians were there in force. And Wayne usually gives away a prize to the person who comes from the furthest away.

The Fix is in!

In response to customer demand, we're going to post CSD details in the newsletter on a regular basis. At time of writing, the most recent CSDs available are one for the compiler (number 9, formal service) and WorkFrame/2 ™ (number 2, informal service).[10]

Compiler fixes

CTC0009 contains the following fixes, common to C Set ++ V2.0 and V2.1:

Fix/APAR/Component	**Problem description**
PJ13703 - CFE	Defining the first token in a program as a character string with the /D option causes an internal compiler error
QUAL-IMP - CFE	Label not found message does not indicate which label is not found.
QUAL-IMP	Internal compiler error while generating /Wenu messages for program with 'case' statement outside 'switch' scope.
PJ13705 - BE	Fixed a parameter passing problem that occurs when the last statement of a function is to a _Pascal function.

[9] SmallTalk is a trademark of Digitalk Corp

[10] In the next issue, we'll carry the story of how CSDs are named, and all the places they can be found.

QUAL-IMP - BE	Allow the optimizer to assign registers a little more efficiently in certain circumstances.
PJ13706 - BE	A loop that contains a goto to a return could, in rare cases, cause incorrect code generation.
PJ12329 - CRT	A 16-bit function will get a protection exception if the function has a large automatic storage area. This problem occurs on secondary threads, and happens because the stack is not committed when the thread begins execution and the 16-bit code does not have stack probes. The compiler runtime code that sets up the 16bit stack needs to commit it.
PJ13704 - CRT	Compiling 16-32 callback funtions with static link results in duplicate references when linking. Reference APAR PJ12507
QUAL-IMP - CRT	Now checking if a destructor exists during error recovery in __vec__copy(). May have caused traps

Workframe fixes

Wf21_2 contains the following fixes (applicable to V2.1 only):

PTR	**Problem description**
22297	Base project template does not recognize the changes made to its default settings so that when a base project is created, settings revert to the default.
22302	Base projects lose their icon view settings.
22333	Closing the base project Settings notebook on the Target page sometimes locks the system.
23330	Double-clicking on file names to invoke EPM does not work as expected when file names are unqualified, or when they are mixed- or lower-cased.
22300	Start button not disabled while make file is being generated.
22330	MakeMake doesn't have the C Runtime messages bound to it, so when it hangs, no register information is shown.

22218	The IBM Library Manager fails when duplicate object file symbols are found while combining extremely large libraries.
22225	The IBM Library Manager outputs an erroneous warning "Duplicate symbol...ignored" when an input library contains an exported name (EXPDEF) that is an alias for an internal (PUBDEF) name.
22326 and 22324	When parsing resource files for dependency information, DDE3DEF2.DLL does not remove the optional quotes from filenames or respect comments prefixed by "//".
22316	When parsing resource files, DDE3DEF2.DLL finds include files where it shouldn't.
23912	SomMethodDebug calls have not been removed from the source.
22314	Mask matching fails in some cases for HPFS file names.
22329	The WorkFrame/2 program makes an invalid assumption about variable string formats that causes MakeMake to hang and the Monitor to trap when actions are invoked.

A complete list of fixes (i.e. all fixes since GA) is given in the LIST file that comes with every CSD file.

Advice

If you are running with CTC0008 applied, we recommend moving to CTC0009 if you are using EXTRA (APAR PJ13704 fixes a problem which surfaced with CTC0008).

If you are using Workframe/2 at GA level (i.e. you haven't applied CSD 1), we recommend you apply CSD 2; it has fixes in it you ought to have. CSDs are cumulative, so getting CSD 2 automatically gives you CSD 1. If you've applied CSD 1, CSD 2 is a nice-to-have.

Other fixes

ICLUI, the UI class library, has fixes available:

- CTL0005 for C Set ++ V2.0
- CTM0005 for C Set ++ V2.1

The utilities grouping (IPMD, Browser, EXTRA) has CTU0002 available.

Roger Says...

It's time

A user recently asked about setting GMT.

I have to set date and time on a PS2 at the reception of a string giving the GMT time/date. What C instruction should I code to enter the GMT into the PS2, knowing that the DosSetDateTime can set a LOCAL time and not a GMT time?

My environment var TZ follows the European rule for DST like
TZ= DDT-1UNO,3,4,0,7200,9,4,0,10800,3600

(DDT and UNO are dummy names, other values are actual ones for France)

"1997/12/25-08:13:45Z"
(self explanatory string gives GMT date/time)

_tzset() allows my process to take TZ into account

This is either easy or hard, depending on whether the OS keeps local time or GMT.

1. Assuming that the internal clock on the machine keeps GMT, this is a "no brainer". Use the GMT time string to fill in the parameters for a DosSetDateTime call.
2. If the internal clock on the machine keeps local time, things are a little messy, since there are no functions to convert from GMT. The following rather ugly code should work:

```
#include < time.h>

struct tm * lcl_t;  /* local time */
time_t t;           /* GMT time */
struct tm gmt;
static char bfr. 256‘;  /* MUST be static! */
int i;

/* load the GMT time string into a struct tm */
loadtm(&gmt, timestring);

/* tell the program (temporarily) that we are on GMT */
i =  sprintf(bfr,"TZ= ");
strcpy(bfr+ i,getenv("TZ"));
_putenv("TZ= GMT");
_tzset();

/* get the corresponding time_t value */
t =  mktime(&gmt);

/* and return the program to local time */
_putenv(bfr);
_tzset();

/* now get the corresponding local time */
lcl_t =  localtime(&t);

/* now you can fill in the parameters for the DosSetDateTime() call
   from the lcl_t structure */
```

I quote

To define a preprocessor variable on the command line that contains quotes, spaces or other special characters, use the following syntax:

```
/DPPVAR= "\"this is a \\  \'preprocessor\n string\'...\""
```

This is the equivalent of the following line in source:

```
#define PPVAR "this is a \\  \'preprocessor\n string\'..."
```

Can you get there from here?

Migrating your code from an 16 bit compiler to the C Set + + compiler can be an educational process. You will probably discover a number of latent bugs in your code (the "how on earth did that *ever* work!" variety).

If you just compile your code with the C Set + + compiler, its much tighter type checking will probably present you with an intimidating list of errors and warnings. Here's a methodology that will probably assist:

1. Rebuild your application with your existing compiler, but turn on all available warnings. Go through every compile message, and fix the code to prevent the message. This will probably turn up a few bugs. Especially important - fix all missing function prototype messages by including the appropriate header.

2. Check the code over for the use of variables of type "int". In many cases you may find that "short" is what was really intended. If so, change the declaration to "short".

3. Since you have now walked through the code, you are in a better position to decide whether a common code base between 16 and 32 bit code is desirable or practical. This determines whether you need to make any further changes using conditional compilation. You can use the predefined macro __IBMC__ to determine if you are compiling with C Set + + .

4. Where OS/2 APIs are used, change the variables from USHORT to ULONG, and SHORT to LONG, as the 32 bit APIs take 32 bit parameters.

5. If calls to subsytems that are to remain as 16 bit are present, change the function declarations to reflect this.

6. Compile under C Set + + , using the /Wpro, /Ti, /O-, and /Sm compile flags. Fix up warnings and errors.

7. Run your application. You may have some problems. Use IPMD to locate them (that's why you compiled with /Ti and /O-).

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.

As some questions are posed by more than one user, you may notice "your" inquiry worded slightly differently. We've probably consolidated similar queries that can share a detailed answer. If you feel we haven't quite got your point, let us know, and we'll go around again.

Question	**Answer**
Q: CSD install problem I get an abend in DOSCALL1 when running SERVICE to install the CSDs for C Set + + Help!!	A: This problem occurs when trying to apply the CSD to a copy of the compiler located on a Novell LAN server. It's caused by a delayed file close on the LAN server, which prevents the service program from reopening the file. The workaround is to XCOPY the compiler onto an OS/2 machine, apply the fix there, and then copy it back to the server. If you want to minimize the number of files copied, you only need copy the following files: IBMCPP\SYSLEVEL.CT* IBMCPP\DDE4FIX.LOG IBMCPP\DLL\DDE4FIX.DLL And all the files listed in the SERVICE.LST file of the CSD. Assuming the copy is in directory D:\toolpath, you apply the CSD with the following command SERVICE D:\path
Q: Workframe error msg I've installed WF V2.1. Whenever I open the templates folder, a window titled "Action Log - History" pops up containing the message "Unable to load action profile < default action profile> ". How do I get rid of this?	A: This is a side effect of changing the name of the default actions profile file. You can do this unintentionally by moving or renaming the Workframe/2 folder. To recover, the easiest way is to find the default actions profile (it started out in the "Actions Profiles" folder in the "IBM Workframe/2 2.1" folder), and then open and close it. This causes it to advertise its current location, so that all projects can find it.
Q: KASE:Set and KASE:VIP When compared to KASE:VIP, KASE:Set seems to lack function. Is it complete in itself?	A: Kase:Set allows you to design the GUI portion of your application (windows, menus, dialogs with CUA 91 controls, notebooks). the code generated can be C using the native PM API, or C+ + using ICLUI (UICL). KASE:Set is not demo code - the code produced is complete, and any you can compile and link any programs you produce.
Q: Installing from the CD I bought C Set + + V2.1, part number 82G3732, and I can't find the compiler on the CD that came with the package. The Workframe's not there either. What's happened to it.	A: You can't find it because it's not there! The package you bought is the 3.5" diskette version. The package contains a CD but it's the MMPM/2 (Multimedia) CD, not the entire C Set + + product. We ship MMPM/2 because it's part of the Toolkit which is part of C Set + + . So you end up with a CD ROM in a diskette version of our product.

Question	**Answer**
Q: strupr() and stricmp() The strupr() and stricmp() library functions don't work properly when they are used in a DLL. They work fine when only a single copy of the DLL is loaded but they return incorrect values when used by any additional processes which load another copy of the DLL. strupr() always returns a zero-length string and stricmp() and memicmp() always says the two strings/memory are equal. These functions don't exhibit this behavior when they are used in an EXE and multiple copies of the EXE are running. Is this me or C Set + + ? Am I missing a CSD?	A: Looks like your DLL is missing either INITINSTANCE TERMINSTANCE or DATA MULTIPLE NONSHARED.
Q: Inline warnings, lack of The C Set + + compiler does not give any warnings when a function that has been requested as "inline" is NOT made inline. Is this correct, and if so, are there plans in the future to provide some kind of warning message? It might be nice to see some information on this in the map file. i.e. Function name and how many times it was inlined, and how many times duplicates were removed.	A: Yes, putting out a message saying that an inline function has not been inlined is certainly something we're aware of for the next release
Q: Exception with O+ compile option My program was compiling fine without the O+ compile option specified. Now that I added that option in, I get the following exception. How do I fix it? Exception #C0000005:8E461 occurred in scr2draw.c function scr_draw().	A: The trap will disappear once you install the CSD CTC0009.
Q: C/C+ + Tools Actions Profile for WF 2.1 I've installed C Set + + and WF 2.1 on two different machines. On one, the C/C+ + Tools Actions Profile contains COMPILE, DEBUG and LINK actions. The other contains these, plus a number of ANALYZE and one BROWSE action. Why are the two different? It's the same product.	A: When you installed a second time, you didn't install EXTRA and BROWSER, so you don't see these two actions in your action profile.
Q: Collecting trace information from dynamically loaded DLL's How can I collect trace information from DLL's which my program loads dynamically at run-time. I have compiled the DLL's with /Gh+ and linked them with DDE4XTRA, but I cannot seem to get any trace information from them.	A: Currently, we can not trace dynamically loaded DLL's. In order for Extra to trace calls in a DLL, it must be preloaded. There has been a lot of need for this feature, and we have added it to our growing list of future enhancements.

Question	Answer
Q: C Set + + templates Is there any way to use templates without invoking the linker through icc?	A: No documented or supported way.
Q: Pascal calling sequence Does C Set + + support the "Pascal calling sequence."	A: C Set/2 and C Set + + support both 32-bit and 16-bit Pascal caling conventions.
Q: Inadequate error handling when /Sv is omitted I have a C application using memory files which used to be compiled with C Set/2 and worked fine. After rebuilding with C Set + + all of a sudden memory files didn't work. I found the new /Sv option which must be specified when compiling programs which use memory files. Now the application is working again. My complaint is that the error handling appears to be broken in this area. When I call 'fopen' to open a memory file (when compiled without /Sv), 'fopen' returns a NULL pointer indicating a failure. When I query 'errno' to find out why, I either get zero or a value from some previous library function call, which means that I never get useful information about the memory file failure. This made it much more difficult to figure out what was wrong. The 'errno' value looks like a bug to me. Please tell me more about /Sv.	A: The /Sv option was added to save EXE size. Basically, if an application do not use memory file, we will not pull in the code (using /Sv). It cannot be detected at compile time whether you will be using memory file, we added this switch to cause the "real" memory file code to be pulled in. Without /Sv, a dummy function is pulled in to satisfy the external referenced by the runtime. I do agree with your view that setting the errno will be a good thing and it should be done (we'll add it to the list). But I wouldn't say it is a bug since ANSI doesn't require ERRNO to be set for fopen() in the first place.
Q: Conversion from MicroSoft C V6.0 I am working with a commercial API package which has given me two .h files with information for compiling with Microsoft 6.0. I am trying to compile this under C Set + + instead. When I compile their sample application I get com-plaints about three items (all having to do with linkage of variable definition). The items are: _far * which I converted to * _Seg16 _pascal which I converted to _Pascal _loadds which I don't know what to do with While I was able to compile and link with these changes, the sample application will not run. Are my conversion correct? Is there anything else that I should look for?	A: The conversions should be: _far * to * _Seg16 _pascal to _Far16 _Pascal _loadds to nothing. There is no equivalent. Also, if you are calling these functions, you should use the /Gt compile option to insure that all objects are tiled. Also beware of ints, which are 16 bytes in size with MSC, but 32 bits with C Set + + . Explicitly use short (16 bits) and long (32 bits). _pascal should become _Far16 _Pascal , and _loadds can be deleted.

Question	Answer
Q: ICC response file format Does the ICC response file format require any special characters (like pluses between elements) and can it list OBJ's instead of CPP's. What about OBJ's and CPP's out on that TEMPINC thing? Will it find those? Will it make them too? Can I also list libraries, map files, definition files, etc...?	A: Carriage returns in the response file are treated as white space. There are no special line continuation characters. The file is parsed as though the whole file was one big command line.
Q: Why does CSet + + / WF use ICC to LINK ? Why does workframe produce a makefile that uses ICC to invoke link. Doesn't ICC in turn call LINK386 anyway ??? Why does it not just use the LINK386 statement instead of the crypted ICC statement	A: Because linking C+ + objects requires more than a call to LINK386 in some cases ... linking a C+ + program requires that the compiler compile the code for any templates that are used. This compilation occurs during the link step. If you link with LINK386 directly, this compilation does not occur and the link will fail.
I understand but that's C+ + There isn't a difference for C is there?	C does not *need* to link via ICC, but the linker dialogs under workframe don't know that you are linking straight C code, and make the pessimistic assumption... Alternatively, you can use the linker dialog from the WorkFrame to link straight C codes.
Q: Order of Execution of Statics in DLLs I have tried the pragma priority and have had success for statics within the same .exe (different .obj that are NOT in DLLs). When the .objs are in different DLLs I can't force the order with the pragma priority. Is there something else that supersedes the pragma priority? (Is there a tool that lists the order for the statics for a DLL and an .obj? There must be something in each .obj header so that the pragma priority for each .obj can be ranked for which .obj goes first in the link process.)	A: The ordering of static object initialization between DLLs is the same as the order that DLL initialization takes place. The rule that applies here is that no DLL will be initialized until all DLLs that it uses are initialized. OS/2 2.1 enforces this, and refuses to start up an application that has a loop in its DLL initialization..
Q: Debugging a DLL used by a rexx program I need to debug a DLL that is coded with C Set + + and that is used by a rexx program.	A: IPMD will allow you to debug the dll. Assuming your REXX command is r.cmd, and your dll is mydll.dll, the following will allow you to debug your DLL: 1. Start IPMD like this: `IPMD CMD.EXE /K < your REXX.cmd>` 2. When IPMD displays the disassembled code for cmd.exe, set a load type breakpoint to stop when your DLL is loaded. 3. Now run under the debugger. It will break as your DLL is loaded, and you can then set breakpoints in your DLL.

Question	**Answer**
Q: Exception on calling DosWaitThread How can I avoid the exception: Exception = 80010001 occurred at EIP = 100b It happens when I call DosWaitThread with the wait option.	A: That's an "unable to grow stack" exception. You need a bigger stack! How much stack space does your calling thread and the thread you are waiting for have? Also, are you using /Wpro to make sure that DosWaitThread is being prototyped properly?

Q: ICC gives U1095 'command line too long'
 	
I get error U1095 'command line too long' when trying to link a lot of files together. What is issuing this message? IS it ICC? How do I avoid this problem? I've tried specify icc @input.file with input.file containing the ICC parameters but haven't gotten that to work either (DDE4MNCH: could not open file "\.obj"). Is specifying the @file the right approach or is there some other way to get ICC to take a longer input string?
`NMAKE : fatal error U1095: expanded command line 'icc /Tdp /Wgen /Gm+`
` /Ge- /B" /NOI /NOE " /Fe ftbsys.dll file1.obj file2.obj`
` ...`
` file100.obj file101.obj' too long.`
	A: NMAKE and CMD.EXE have a line length limitation (it's approx. 1023 bytes). The response file is the correct approach. Note that an ICC response file has no line continuation characters....

Q: Deadlock inside free() when passed debug allocated memory
 	
We had a bug in our code where we were freeing memory in our dll that was allocated by another dll using debug_alloc. Both dll's had the runtime statically linked, and were compiled with multi-thread libraries. Our application would hang, and would require a re-boot in order to kill it and unload our dll's.
 	
Using the Kernel Debug, we found the following stack trace:
 	
`##.k`
`005b:11305dc5 11183a1d 11321548 111e0d88 00027f68 malloc + 29`
`005b:11306f70 00027f28 00000000 005f3850 006dbea0 _Regen_exception`
`005b:11309d7d 00000001 11321548 113216a0 00027f80 _fprintfieee + 31`
`005b:11305fa6 00000000 113216a0 11305ead 00000000 _Weak_FreeListEntry + 12`
`005b:11305f52 006f3290 00000000 00028aa0 11306d4c free + 4e`
`005b:1130302c 00809490 11310208 00027fce 00027fd4 Store_MenuItem + 270`
` .<_extrastuf>`
`005b:000102b4 00570224 0002899c 0002888c 00028894 main + 2b4`
`005b:000103b9 00000002 00570210 00570010 ffffffff __RunExitList + a1`
`005b:1a0302af 0000073c 00000000 00030000 00030dd9`
`##`
 	
Further looking found us on a RequestSem inside of malloc. It appeared to be the same sem that was successfully requested inside of free(). Our speculation was that the sem should have been released before calling Weak_FreeListEntry, or one of the lower level calls.
 	
What's wrong?
	A: Your problem is actually that you are allocating the memory using one copy of the library, and freeing it with a different copy of the library. Since the library code has data areas associated with it (the library environment), it gets confused when data from another environment is freed. Try dynamically linking to the library.

Question	Answer
Q: calling __ctordtorTerm() and _CRT_term() from _DLL_InitTerm() If I write a C++ DLL with my own _DLL_InitTerm() function, I understand I have to call _CRT_init() and __ctordtorInit() when my DLL is loaded. The Programming Guide also says I have to call __ctordtorTerm() before calling _CRT_term() when the DLL instance is terminating. From the example in the Programming Guide, if I'm dynamically linking to the run time library, I do not need to call _CRT_term(). Is this right? My question is: If I'm dynamically linking to the runtime libraries, do I need to call __ctordtorTerm()? If so, I take it I must register an exit routine from which to call it, right?	A: If you dynamically link to the runtime libraries, they will take care of their own initialization and termination, so you should not need to call _CRT_Init and _CRT_Term. However, you do need to call _ctordtorInit and _ctordtorTerm in your DLL initialization routine to take care of any static C++ objects you may have. The default DLL init/term routine does this for you. You may call _CRT_Init(), since the runtime library protects itself against multiple initialization calls. Doing so also insures that you are independent of DLL initialization order. If you are dynamically linking to the runtime, you cannot call _CRT_Term. The library does not export it.
Q: Global Ctor/Dtor We have a global object that creates a couple of threads in the constructor. In the destructor, we synchronize the take down of these threads so that certain information may be stored away. It seems that prior to the global dtor being run, all threads are killed. We know of a couple of workarounds for this behavior. But I was curious why all threads are killed prior to running dtors.	A: Global destructors are run either in a DosExitList or as part of DLL termination. In both cases, all threads have been terminated first by the operating system.
Q: NLS and Double-Byte Support Does C Set++ have NLS and Double-Byte Support? If yes, what would be required to enable an application written today for NLS and Double-Byte support	A: C Set++ provides the ANSI standard multibyte string calls, and supports input and output formatting of multibyte strings in the scanf() and printf() library functions.
Q: Calling C++ from C Is there a way to call a C++ function (not necessarily a member function) from a C function ?	A: As long as the C++ function has been defined as extern "C", it uses one of the linkages compatible with C, and can be called from C. Needless to say, this linkage is not available for a member function..
Q: _Optlink problem I wrote a simple program explicitly specifying _Optlink for a function with a variable argument list. The compiler is still pushing arguments onto the stack. Is this a compiler bug?	A: This is not a bug. Take the printf() function, for example. Its declaration is `int _Optlink printf(const char *,....);` This function will always pass its first argument in the EAX register, like any other _Optlink function. However, the _Optlink convention requires that variable parameters be passed on the stack, so the second and third arguments are not passed in the registers EDX and ECX, but are pushed onto the stack.

Question	Answer

Q: Exporting C++ member functions in a DLL
 	
What is the proper/recommended way to export C++ member functions in a DLL? _export etc. is invalid in the function prototype. It appears that the only way is to have a standard C function as a wrapper to the C++ class function.
	A: You can specify the _Export keyword either in the class definition or in the function definition, e.g.
	
	`class A {`
	`   void _Export fun1();`
	`};`
	
	`void _Export A::fun1() {`
	`    return;`
	`}`

A Word from your Editor

This newsletter is available in softcopy. You'll find us in several places:

- OS2BBS of IBMLink
- Section 5 of CompuServe OS2DF1
- IBM PC Company's OS2BBS
- IBM's EMEA DAP network
- posted, with permission, on several other nets.

Please contact us if you have any questions about posting electronically.

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!

This newsletter is available softcopy on several networks. If you obtained your copy electronically, you may not be on the C Set ++ mailing list. If you're not, you're missing out! You're not receiving the product information, information updates, and other general mailouts that others are. You can be on our list for these goodies, with or without hardcopy newsletter distribution. Just mail/fax your full address (phone number is handy too) using the Reader's Reply Form at the back of this copy, and circle whether you want the NL hardcopy, or just want to be added to our mailing list.

This newsletter was produced by the C+ + Service & Support department of the IBM Software Solutions Toronto Lab. For further information on any of the products mentioned, please contact your local IBM office, or an authorized IBM Business Partner.

Numerous product references in this publication are registered trademarks or trademarks of International Business Machines Corporation. IBM Canada Ltd., a related company, is a registered user.

Sample code is provided for information purposes only, and is used by readers at their risk. IBM makes an effort to provide accurate and safe code examples but does not warrant their correctness.

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.

This newsletter is © IBM Corporation 1994.

IBM C Set + + News No.7 June 94 issue

Reader's Reply 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?*
4. *If you received your copy electronically*
   - *would you prefer to receive hardcopy?*
   - *would you like to be on our mailing list only (no hardcopy newsletter)*

   (circle one)

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. Please mark your fax for the attention of MJ Houghton.

IBM C Set ++ News No.7 June 94 issue

Reader's Reply Form

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

IBM

Software Solutions Toronto Lab
IBM Canada Ltd
C Set ++ Service & Support, Dept 812
22/812/844/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 and tape......fold