PowerBASIC Gazette

We put the POWER in BASIC!

Volume 3 Issue 1

32-bit Text Mode Apps for Windows!

Tired of the hassle of "drag and drop" form designers? Lost in a sea of API calls just to display text and numbers in a window? With the PowerBASIC Console Compiler, it's a whole new Windows!

Compile Basic code to sizzling text mode programs. For Win95, Win98, even WinNT. The "console" is a text mode interface connected right to the heart of 32-bit Windows. No fluff. No animated puppets. Just intense computing power. Put your programming effort where it belongs – at the core of your code. And when your calculations are complete, it's a simple matter to PRINT, or even LPRINT, the results.

For years, DOS programmers have been looking for ways to extend the capabilities of their programs. Struggling with limited memory and competing standards. Nearly every computer has XMS, but it's slower than EMS. And then you're usually limited to 16 megabytes or less. 32-bit Windows means that memory is measured in gigabytes, not kilobytes or even megabytes. And you don't need to use special keywords to create a multi-megabyte array. Just DIM it as you usually do.

You've written code for storing your data in files but how do you import from a Microsoft Access database? Fortunately, any Windows program can access the ODBC API for reading and writing data from nearly any data source. SQL, Oracle, FoxPro, Excel, whatever the source, if you have an ODBC driver, you can access your data.

Usually a brand new compiler means months of waiting for vendors to supply add-on tools. But PB/CC can access any standard 32-bit Windows DLL. That means you can choose from thousands of existing libraries. Serial communications, TCP/IP communications (Winsock), SMTP and POP3 for internet email, and even form engines for quickly creating data entry screens.

We didn't just make a text mode Windows compiler. We created a compiler that allows you to really take advantage of 32-bit Windows programming. Unlimited memory, Multi-threading, Register Variables, Matrix Operations, and access to the entire Win32 API including the Winsock and ODBC API's.

And for those of us who are writing applications for Windows 95/98 and Windows NT web servers, PB/CC eliminates the need for slow interpreted Perl scripts. By directly supporting STDIN and STDOUT, PB/CC can interface directly with the Common Gateway Interface on your web server. And not the hybrid WinCGI, but true CGI.

CGI applications written in Visual Basic require at least 3 megabytes of memory for each instance. Any more than a hundred simultaneous accesses to your server can result in delays and memory errors. With PB/CC a typical CGI applications uses less than 80k of memory, allowing thousands of simultaneous accesses to your web server. And unlike ISAPI applications, your CGI applications completely unload themselves from memory when they're finished. So your server doesn't end up with hundreds of little used Web applets clogging up memory and slowing down web server performance.

PB/CC is available from PowerBASIC for $149 plus shipping. It requires Windows 95, 98 or NT, two megabytes of hard drive space and 4 megabytes of RAM.■

PB/DLL 5.0 gives you 23 times the performance of Visual Basic 6!

The PowerBASIC DLL Compiler for Windows is now available in 32-bit It's not interpreted, and there's no thunking, just sizzling fast 32-bit machine code.

PB/DLL 5.0 compiles Basic source code into true 32-bit EXEs or DLLs for Windows 95, 98 and Windows NT. A typical Sub or Function converted from Visual Basic will run 8 to 10 times faster in a DLL. With a little optimization in the code to take advantage of Register variables and optimizing your variable types you can easily achieve 23 times the performance or more!

PB/DLL 5.0 is not just a 32-bit port of our 16-bit compiler. It's been completely re-written to use every feature of Intel's 32-bit CPU and the 32-bit Windows environment.

We've got *Threads*… With PB/DLL, you get complete support for multi-threaded apps, even with Visual Basic 6. Every aspect of this new version supports code reentrancy — a

(Continued on page 3)

what's inside...

PowerBASIC on the Net – It's where you want to be today!

"The Information Super Highway", "the I-Way", "Cyberspace", whatever you choose to call it, the internet has quickly become the most important tool for the survival of today's programmers. And PowerBASIC on the Net has become a required daily stop for any Basic programmer looking for an edge.

PowerBASIC.Com has grown from a marketing tool into a major resource of programming examples, answers to frequently asked questions, and a meeting place for Basic programmers from all over the globe.

Example Code

From our web site, you can download all of the example code from any PowerBASIC compiler product. Including any updates to existing examples and updates to the online help files.

You will also find many new examples and files which were created by users like yourself. Choose from hundreds of public domain, freeware and shareware files.

Support

Our Peer Support Forum allows Basic programmers to post questions about the programs they are writing, answer questions that others have posted or simply read the existing messages. You will also find answers to many frequently asked questions, posted by our support staff and marketing team. The new forum software is easy to use and allows you to search through all messages past and present for items of interest. And the new "Source Code" forum lets users share code for use by everyone. Need a fast function to reverse the contents of a string? You'll find it here, along with hundreds of useful code snippets.

Online Shopping

And, we've expanded our Online Store with a new Secure Server for real-time order processing. Now you can order products online and download many of them once your purchase has been made. No more waiting for disks and packaging to be shipped to you. Of course, if you prefer to have a disk and printed manual shipped to you, we can still do that too! Shopping online has never been easier.

Online Resources

If you're looking for third-party tools or other resources for your programming needs, PowerBASIC.Com has links to every PowerBASIC related site on the internet.

Stop by the new Usenet newsgroup **comp.lang.basic.powerbasic** and say hello. (A big thanks to Marc Van Den Dikkenberg who was instrumental in creating the new newsgroup.) You'll find many great tips on using all versions of PowerBASIC, source code for your collection, code optimization tips and more.

Electronic Newsletter

And don't miss out on the most valuable resource on the internet – The PowerBASIC Gazette — Electronic Edition.

This weekly e-mail newsletter contains articles on advanced programming, tips and tricks, third-party add-on reviews, and insight into the future of Basic programming. Even the future of PowerBASIC itself. If you're not a subscriber, just drop a subscription request with your name and email address to **email@powerbasic.com** and put some *Power* in your *BASIC* programming today! ■

Dave's World – Living on the Edge

Looking back on the past year, I can hardly believe how exciting things could get in only a twelve month period. PowerBASIC released a couple of true 32-bit compilers for Windows, a DOS compiler update, a B-Tree index manager, and a really cool Windows GUI App design tool (my favorite).

We wrote so much source code that I'll bet our entire development staff wore out two or three keyboards each. Some of them were typing so fast at times, I could swear I saw smoke coming from their fingers. <smile>

Our support staff has been pretty busy too. If you've been on the web site lately, you've seen the new Web BBS software. No longer do you need to download huge archive files and search them offline. All messages are kept on the new BBS and it has a built-in search engine.

On a more personal note, before we moved into our new building (without a basketball goal <sob>), we had a little free-throw shooting contest in honor of the Bulls winning another championship. It was a lot of fun, with many of us huffing and puffing as we chased "air balls" down the hill. I did pretty well, although I'm much better at lay-ups than free throws. Bob won the contest, but nobody let him win. He's actually pretty good. See ya on the Net! ■

New Telephone Numbers!

Carmel, California has changed its area code from 408 to 831. However, our basic phone numbers have not changed.

Call PowerBASIC Toll Free at	**(800) 780-7707**
Outside the U.S. call	**(831) 659-8000**
Customer Service	**(831) 659-8000**
Fax	**(831) 659-8008**

You can also reach us on the Internet:

World Wide Web	**www.powerbasic.com**
Sales	**sales@powerbasic.com**
Support	**support@powerbasic.com**

Don't forget to give us your email address so that we can send you discount offers and add your subscription to the electronic version of the PowerBASIC Gazette. Send your email address to:

email@powerbasic.com

PowerBASIC Gazette is a creation of the PowerBASIC, Inc. product development and marketing staff. It's purpose is to inform customers and prospective customers of new developments and tips for using existing products.

(Continued from page 1)
must for modern operating systems. And with PB/DLL, multiple threads can even be launched from VB applications which lack this vital functionality.

We've got *MMX Support*… With PB/DLL, you get complete support for these new primitives. Plus, the inline assembler offers floating point, 80486, and Pentium opcodes, too.

We've got *Register Variables*… In each PowerBASIC function, declare up to six unique register variables. Integer class or floats, the choice is yours. These six special variables are stored in hardware CPU registers, not conventional memory. Since register access can be five times faster, just think of the boost you'll get! Best of all, you the programmer can choose the optimization which is crucial to your application. Register variables are the ***ultimate optimizer***.

We've got *Matrix Operations*… PowerBASIC is the first mainstream Basic to offer full matrix support. Initialization, assignment, scalar and matrix math. Identity, transposition, even inversion. They're all there. And at sizzling performance levels.

We've got OLE Strings… For compatibility, for efficiency, there's OLE2 support. Swap dynamic strings directly with Visual Basic or other languages. Direct import and export of string functions and parameters. It's all there, for all your string needs.

We've got a *Full-Featured IDE*… Includes a multi-windowed source code editor with no limit on files or sizes. Color syntax highlights, context-sensitive help, and a combo-box to quickly jump from one Sub/Function to another. An intrinsic debugger with full executable support. Even automatic compilation and embedding of industry-standard resources.

We put the POWER in BASIC… We've said it before, but it has *never* been more true than it is today. With version 5, there's a brand new ***War On Bloatware!*** For the very first time, it's feasible to deliver a 32-bit Windows application on a single floppy disk.

We put the POWER in Visual Basic, too… Use Visual Basic for what it does best, user interface code and forms design. Use PowerBASIC for what it does best, number crunching, calculations, your mission-critical code. No VB programmer should be without PowerBASIC.

Of course, there's even more, like pointers, unsigned integers, unions, and multiple currency types. An inline assembler, ASCIIZ strings, and optional function parameters. No other BASIC compiler brings all the POWER of both C and Assembler to your BASIC code.

PB/DLL 5.0 is available from PowerBASIC for $179 plus shipping. It requires Windows 95, 98 or NT, two megabytes of hard drive space and 4 megabytes of RAM. ■

Create complete Windows Apps, it's easy!

Creating a complete Windows application can, at times, be a difficult thing. The "event driven" programming model means that you have to write "callbacks", "event handlers", use "code pointers" and other technical things that most BASIC programmers have never needed to know.

Visual Basic, Delphi and Visual C++ all include "visual design" tools. They attempt to simplify user-interface creation by letting you "drag and drop" from toolbars into windows and dialogs. But you pay a penalty for over-simplification. For example, Visual Basic forms are stored in a proprietary format, not the native format used by Windows. That results in slow screen updates and a general lack of responsiveness in your user-interface. That may be acceptable if your application is used by one or two people, but it's a serious limitation if you need "professionalism". Additionally, all products of this type impose an added burden, requiring that you distribute multi-megabyte run-time DLLs, controls and objects.

Hand-writing the low-level code in Delphi and Visual C++ (and even PB/DLL by itself), will result in a much faster and much smaller application. But it takes much more time (often 5 times the normal cycle) to complete your project. What you need is a tool to visually design your application, using native Windows resources, but which doesn't impose the huge overhead of external run-time libraries.

What you need is PowerGEN!

PowerGEN is a Rapid Application Development (RAD) tool for PB/DLL. In only a few easy steps, you can have the complete user-interface designed and running, typically in under an hour. Just plug in your calculations and data access code to complete the entire project.

Step One, using any Dialog Editor (the Microsoft Dialog Editor is supplied with PB/DLL) you create one or more "windows" with buttons, edit controls, list boxes and any other control you need in your program. Just save it as a standard resource script file.

Step Two, load your resource script into the PowerGEN utility. Select which controls you'd like PowerGEN to automate, or simply select them all. Then watch PowerGEN actually *write* the PB/DLL source code for you.

Step Three, fill in a few empty Subs and Functions with the code you want executed when a button is pressed, a menu item is selected, or any other control event is triggered. And then compile.

In just three easy steps you've created a Windows program complete with a Graphical User Interface. Event-driven code with the professional look you've always wanted.

Code created with PowerGEN is highly efficient, while form loading is virtually instantaneous. It uses binary resources, the native format used by Windows, not a slow, interpreted forms engine. Perhaps best of all, there's never a need for runtime libraries. A complete application can be generated as a single .EXE, often as small as 100K.

PowerGEN supports all of the standard 16-bit and 32-bit controls that are built in to Windows. Including edit boxes, list boxes, combo boxes, scrollbars, buttons, radio-buttons and more. Even the latest controls for these platforms are supported such as property sheets, sliders, spinners, treeview, listview, and progress controls.

PowerGEN is compatible with both the 16-bit and 32-bit versions of PB/DLL. With PowerGEN, the days of slow, fat BloatWare are history. Create state-of-the-art applications for Windows!

PowerGEN 1.0 is priced at $99 plus shipping. It requires Windows 95, 98 or NT, two-megabytes of disk space and 4 megabytes of memory. The 16-bit applications you create will run in Windows 3.1 or later. ■

DOS ain't Dead! And it just got better

We have some exciting news we'd like to share with you... **PowerBASIC 3.5**, the premier DOS programming language, is ready and waiting! And what a compiler it is…

```
Dim Virtual x%(1 to 8000000)
```

An array of eight million integers? In a standard DOS program? With PowerBASIC 3.5, it's now automatic! And frankly, it's that very kind of innovation which keeps PowerBASIC at the leading edge of modern programming tools.

You know, we've been in the compiler business for over eighteen years. We've earned many awards for quality and performance, including ***PC Magazine's Editors' Choice***. We've consistently out-performed the others in execution speed, code size, reliability, and many other important areas. But when you actually try PowerBASIC 3.5 for yourself, we think you'll agree it's the very best DOS compiler ever.

- ***We've got Arrays within Types . . .*** With PB 3.5, User-Defined Types and Unions may now contain member arrays of one or two dimensions. String Ptr is now valid as a member, too.

- ***We've got Virtual Arrays . . .*** Use expansion memory at will -- **PowerBASIC 3.5** handles all the messy details! Allocate virtual arrays in EMS memory, up to 16 megabytes for massive storage needs (some EMS drivers even support 32 megabytes!). Indexes can be huge, as well: Long Integers are supported for both Virtual and Huge arrays. Another convenient benefit... by moving data out of conventional memory, even more room is available for code, a key resource for every DOS programmer.

- ***We've got REDIM PRESERVE . . .*** Re-Dimension an array, yet retain its existing contents. A great memory saver, since arrays sizes can be adjusted to fit current needs.

- ***We've got ASCIIZ Strings . . .*** A new native string type, for compatibility with DOS and other languages, like C and C++. It's a fixed-length string buffer, with a nul used to mark the string end.

- ***We've got Indexed Pointers . . .*** Access data sets and arrays with a single pointer. `@X[0]` addresses the first element, `@X[n%]` addresses the nth element, and so on.

- ***We've got Standard Input & Output . . .*** Add easy file redirection with STDIN, STDOUT, and STDERR. Standard Output can be used for printing to the screen. If re-direction is active, your output is automatically written to a file, and no extra code is needed. CONSIN and CONSOUT can tell you if redirection is active. You can even write DOS programs which communicate with web servers like Personal Web Server for Windows 95. Create web pages from the data in your DOS program -- It's easy!

- ***We've got Integer Class Random Numbers . . .*** While RND returns a float between 0 and 1, RND(1,100) returns an integer between 1 and 100. No more formulas. No more rounding!

- ***We've got Optional Position Parameters . . .*** CVI(x$,6) extracts an integer from position 6 of a string. ASC(x$,3) returns the ASCII code of the third byte. ASC(x$,4) = 0 sets the fourth byte to nul.

- ***We've got even more . . .*** An $ELSEIF metastatement for conditional compiling. The ampersand (&) character now works for string concatenation. FRE(-11) to measure free EMS memory. SIZEOF(var) to learn the maximum length of an ASCIIZ string or User-Defined Type. SETEOF to establish the end-of-file position. TRIM$ to trim spaces from both the left and right sides of a string. ERRCLEAR to clear the error flag and, optionally, return the current error code.

Of course, there's even more, but we think you get the point. ***PowerBASIC 3.5 is Hot***, and if you don't take advantage of this powerful tool, it's quite likely your competitors will. No other DOS programming language can beat it.

PB/DOS 3.5 is available from PowerBASIC for $149 plus shipping. Version 2.x and 3.x owners can upgrade for $59 plus shipping (includes a new 2-volume documentation set). FirstBasic, QuickBasic/PDS and Turbo Basic owners can upgrade for only $99 plus shipping. It requires DOS 3.1 or later, 500k of hard drive space, and 640k of RAM. To take advantage of Virtual arrays you must have an EMS manager such as EMM386.EXE or QEMM installed, with EMS memory enabled. ■

Windows Jargon

For many PowerBASIC programmers, Microsoft Windows is a completely new experience. The 32-bit Windows operating systems, Windows 95, 98 and NT, offer so much to programmers. But wading through all that new jargon can be taxing at times.

32-bit vs. 16-bit

The terms have been used and misused many times, so just what is the difference between 16-bit and 32-bit Windows?

The primary difference is in the way the CPU accesses memory. When a CPU is in "16-bit mode" it divides memory into many 64k chunks, which is the largest block of memory you can access at one time. Why? Because if all 16 bits are turned on in a 'word' the largest number you can have is 65536. That works out to exactly 64k of memory. Unlike DOS, which is limited to 640k because it runs in '8-bit mode', 16-bit Windows can access 16 megabytes of memory.

32-bit Windows runs in '32-bit mode' which means that it can use all 32-bits of a double-word or 'dword'. When you turn them all on, you get 2^32 power or two-gigabytes of memory per 'chunk' instead of just 64k.

Windows & Dialogs

A window is a rectangular area of the screen that acts like a separate independent screen. Each window can be controlled by a different program, allowing you to run and view multiple programs at the same time. Windows come in different sizes, from very small, to a single large window which covers your entire screen. In Windows 95 and 98 programs can create windows which are not rectangular. They can be circles, triangles or any other shape imaginable.

A dialog is a window that contains controls, such as buttons and edit boxes. Typically you will be required to select items, enter one or more values in an edit box, or press an "Ok" or "Cancel" button. A window that pops up with a message and an "Ok" button is an example.

Console Window

A console window is just like a normal window except that it can only contain text, no graphics. When in a "window mode" it can use graphical fonts the same as other windows, but when it is zoomed to full-screen it uses the font built-in to your video card.

Because it can not contain any graphics, it's much easier for Windows to work with and text display output is much faster than in regular windows.

Multi-threading

Multi-tasking is the ability for a computer to run more than one program at the same time. But multi-threading is the ability for a program to run more than one piece of code at the same time. For example, your PowerBASIC program could create a thread which prints a long document in the background while the user continues editing it in the foreground. ■

Applications Need Some Juice? Tap PowerTree!

Almost every useful program needs a data file of some kind. In fact, in many cases, the data is the entire reason for the existence of the program! But a key requirement is the need to find the specific data you need, quickly and accurately. You may need ten million customer names in alphabetical order... many mailers in zip code sequence... perhaps help to find an employee with a name that sounds like Smith (or could it be Schmidt, or Smit, or Smythe)... or even the urgent need to catalog and count all the words in the latest report from the Independent Prosecutor? All of these common problems can quickly benefit by applying the concepts of a good Index Manager.

Just in case you're a tiny bit "rusty" with all of these terms, let's briefly describe them: An Index Manager is a general purpose library of functions, used to store and retrieve data quickly and efficiently. It does this by creating an index (a map, so to speak) of the important elements of the data, and information on where the data may be found. The index and the data may be intertwined in a single file, or they may be logically separated. There may be one index per data file, or many of them. Perhaps one index to yet another index. You might even create an index with no data at all! The real point here is fundamental... with a truly versatile Index Manager, the sky is the limit.

Generally speaking, each entry in the index contains a "Key" and a "Pointer". The Key is usually a string of something less than 100 bytes, and the Pointer is typically a 32-bit long integer. The secret is to use this simple information to manage fast access to your data in just the manner you need it. Not in a fashion dictated by the authors, but in just the manner and form you desire, to optimize the performance of your application code.

Let's take a simple mailing list of ten million customers. You might wish to access them by name, the sound of the name (more about soundex codes later), or the zip code. So just create three indexes. For every customer, add an entry to each index with keys like "KALE", "K400", and "95066". But all would use the same pointer, the record number of Mr. Kale's main data record. Later, by using the other functions, you can retrieve your data sequentially, by exact match, by approximate match, or even by a sound-alike algorithm.

Just in case you hadn't already guessed it, PowerBASIC Inc. recently released PowerTREE... a very efficient Index Manager, with versions for DOS or Windows (both 16-bit and 32-bit). When the PowerTREE project was first conceived, the planners laid out a number of fairly lofty goals: It must be exceedingly fast, affordable, and use just a very small memory footprint. It must be based upon the prized B+Tree algorithm, allow duplicate keys, and must support networked, even multi-threaded applications. It must offer very large key size, and work well with virtually any format of data file. We think we've succeeded on every count. Read on a bit, and see if you agree...

Binary Trees, B-Trees, and B+Trees

Most every indexing scheme is based upon a tree structure, the first and oldest being the Binary Tree. In this scenario, each block of the index contains one entry, along with information about the next greater and next lesser blocks. Searching or traversing a binary tree is, in effect, a binary search -- much faster than scanning the file sequentially. You start at the mid-point, and continually cut the remainder of the file in half until you reach your target. However, with this small block size, it's clear that one would still do a great deal of "churning" from block to block to block in order to find a desired record.

Binary Trees quickly gave way to the classic B-Tree algorithm, since larger block sizes typically yield much better efficiency. In a classical B-tree, multiple keys and pointers are stored in blocks. To see how it works, consider a block size of 64 bytes. In it, we want to store a key of 14 bytes and a long integer pointer (we'll call the pointer the record number, but it can be anything... even a code number or a byte position in a text file). So, in this case, each key-plus-pointer needs 18 bytes of storage, meaning we can store just three entries per block. That's a total of 54 bytes with 10 bytes left over. Not enough for another complete entry, so the last 10 bytes represent just so much wasted space.

When you add a new entry to a B-tree, the indexing package must search the blocks to find the appropriate storage position. The blocks are maintained in alphabetical order, and a binary-search is used to find the one spot where the new entry should be inserted. If a block becomes full (that is, no room to insert another entry), it must be "split" to create additional storage space. In the first algorithm, a block was split by simply shifting all higher blocks up one position in memory, creating a "hole" for the new block. In essence, the old block became two blocks, and the new entry could be inserted in the appropriate position. Not terribly efficient, but fairly simple to implement.

It didn't take long for B-trees to be enhanced. For example, instead of moving all the blocks up during a split, one could just keep track of both the logical position and physical position of each of the blocks. Then, when a split was needed, the new block could be added at the end of the index. Much faster than physically moving all those blocks, but it created a whole new issue: How could the index manager keep a good track on the block sequence? That was essential in order to implement a binary search.

Most B-Tree implementations relied upon a block sequence array. At start-up, the entire index was scanned, to build an in-memory model of the logical sequence. The array was updated each time a block was added or deleted. Fairly fast, but there are obvious problems in a multi-user or multi-threaded environment. While they can certainly be solved, another glaring concern is memory usage: since a separate array is needed for each key, just how much memory can we afford to dedicate to these sequence arrays? There must be a finite limit, or we'd have nothing left for the application! It's for this very reason that early index managers mandated fixed key size, limited pointer size, or even disallowed multiple keys entirely. Index Managers with small block sizes must also limit the maximum key length. PowerTree uses a block size of 4096 bytes, allowing a key size as big as 4082 bytes... Not likely anyone will push that limit too soon.

Over the years, the B-Tree algorithm was enhanced by many, evolving into the state-of-the-art B+Tree. Block pointers were moved to disk, to simplify multi-user access. Then, block pointers were combined into clusters, excluded from ordinary searches, to vastly improve speed through buffering. Backward pointers were added for quicker "Search Previous" duties. If a block was emptied and deleted, it could be removed from the block map record without shuffling others. If an entire block map record was emptied, nearby entries could be borrowed. With this scheme, you could delete 99% of the entries, and still avoid typical index maintenance. Of course, a complete re-index would reduce the size of the index file, but the elegance lies in the fact that it's never a requirement.

So just what is the difference between B-Trees and B+Trees? Actually, that's a very difficult question -- one which seems to have very many answers, depending upon just whom you ask. At PowerBASIC, we use the term B+Tree to just signify the latest version of the algorithm -- one which embodies all of the features we've already described.

Tapping PowerTREE for Yourself...

So enough with the theory -- how about some practical considerations?

PowerTREE is exceedingly affordable. The DOS version sells for just $89.00, and can be linked into any PowerBASIC 3.5 executable. The Windows version includes both 16-bit and 32-bit DLLs in one single package priced at $129.00. And the beauty is that you only buy them once -- No royalties nor run-time fees of any sort. Of course, you can't distribute source.

(Continued from page 5)

You can't distribute the documentation. But you can release all PowerTREE object code to your users, without cost, when it is included as part of another larger software package.

PowerTREE is very, very fast. As much as 31 times faster than some of the most well-known products in the marketplace. Just in case that didn't sink in... up to 3,100% faster than some major players in the database market. In all of our comparative testing thus far, we've found nothing which outperforms PowerTREE. Not a one!

PowerTREE is flexible. There are virtually no limits on the number of concurrent indexes, nor are there any restrictions on the format of your data. PowerTREE manages your indexes... keys up to 4082 bytes and pointers of +-2,147,483,648 and keeps them separate from your data file, which you maintain, to ensure integrity. Simply put, PowerTREE doesn't sabotage your data format -- leave it any way you like: random files with fixed-length records or variable-length records, record numbers indexed to zero or one, dBase DBF files, sequential files, binary files, or even something special you cooked up just for this occasion! Your data, and how it's structured, remains your business. And, duplicate keys are fully supported with reproducible results every time.

PowerTREE is universal. Use PowerTREE with PowerBASIC, PB/DLL, PB/CC, or practically any Windows language (C/C++, Delphi, Visual Basic, or others). PowerTREE uses the industry-standard DLL calling conventions for universal compatibility. The Windows version includes two sets of code, to immediately support both 16-bit and 32-bit versions of the operating system. Best of all, every version of PowerTREE uses the identical file format -- index files can readily be shared between DOS, Win16, and Win32 applications.

PowerTREE is very easy to use. Only 14 functions to learn, not hundreds like some of the others. They're obvious and intuitive... Call one function to create an index, another to add a key entry, a third to delete an item. Search for an exact match, or just the closest match. Search forward or reverse in sequence. It's just that simple and straightforward. How about networks? A piece of cake. Set just one variable, and PowerTREE handles all the multi-user considerations for you! That's right -- regardless of whether your code runs multi-tasking, multi-user, or multi-threaded, PowerTREE does it all. Automatically.

PowerTree for DOS is available for $89 plus shipping and requires PB/DOS 3.5 or later, 300k of disk space for the complete installation. PowerTree for Windows is available for $129 plus shipping and works with any 16-bit or 32-bit programming language that can call standard DLLs, including PB/CC, PB/DLL and Visual Basic and requires 500k of disk space. ■

CGI Programming with PB/CC

As we near the next millennium, it is becoming apparent that more than 50% of all interaction between customers and hi-tech companies will be handled on the internet. For Basic programmers, there's no clear path of migration to the internet. Or at least there wasn't until now. For the first time, Basic programmers can write applications for their web servers without any kind of kludge or third-party interface.

The PowerBASIC Console Compiler for Windows has direct support for 32-bit STDIN and STDOUT. This means your web server communicates directly with your application and your application communicates directly with your web server. Allowing remote users or staff members to access your company's data via the World Wide Web, or even a local intranet, has never been easier.

What is CGI?

CGI stands for Common Gateway Interface, the actual method used by a web server to "talk" with a CGI program. When a web server executes a CGI program, it captures all data sent to and from the "standard I/O" devices.

Using the STDIN keyword in PB/CC you can read the data sent to your program from a web browser form. You can then parse through the data, perform some kind of action (like a database lookup) and then use the STDOUT keyword to send data back to the browser. Typically, you would do this using the HTML language to make it look nice and pretty.

Is that all there is?

Of course, writing a complete CGI program takes a little more than just read and writing data through standard I/O. When you program reads data from a web browser form, it is sent in a format called 'URL encoding'. That means certain characters which can not be sent over the internet (such as a tilde ~) are encoded into a special hexadecimal format that can be transmitted over the internet.

After the input data from the web server has been decoded, you can access the information from the different fields in your form. Each field in your form has a "name" associated with it. And the data for that field has a "value". Together they are called "name/value pairs". For example, you may have a field name called "firstname" and a value associated with it of "bob". The web server will give them to you in the form of "firstname=bob".

Once you have decoded the data into something you can manage and you are ready to begin talking back to the web browser, you must first tell the web browser what type of data you are going to send it. This is done by using the STDOUT statement to print a header. The most typical header is for HTML, so we would print "Content-type: text/html" following by a couple of carriage return/line feed pairs. You can then begin sending your HTML code for the rest of the page.

Because web servers can run on so many different platforms (Linux for example), there are actually several different methods for exchanging data between a web browser and a CGI application. The method described above is called the POST method, and is most common. All known Windows 95 and NT web servers support the POST, GET and COMMAND methods.

PB/CC includes the complete source code to a CGI application and an HTML page with a form that you can test with your web server. It's important to note that you can not call a CGI application directly from a browser, you must call it from an HTML page on your web server.

In addition to the example code, PB/CC includes the code necessary to support all of the CGI communication methods, handle URL decoding, and parsing your name/value pairs into an array that you can more easily access. See the PBCGI.INC file for more details.

For more information on CGI, please visit **http://hoohoo.ncsa.uiuc.edu/cgi/** where you'll find the latest news. You can find more information on HTML and creating forms at **http://www.w3.org**, the standards body of the World Wide Web. ■