The Windows Programmer's Journal Volume 01
Copyright 1993 by Peter J. Davis Number 06
and Mike Wallace June 93

A monthly forum for novice-advanced programmers to share ideas and concepts about programming in the Windows (tm) environment. Each issue is uploaded to the info systems listed below on the first of the month, but made available at the convenience of the sysops, so allow for a couple of days.

You can get in touch with the editors via Internet or Bitnet at:

HJ647C at GWUVM.BITNET or HJ647C at GWUVM.GWU.EDU

CompuServe: 71141 2071

GEnie: P.DAVIS5

or you can send paper mail to:
Windows Programmer's Journal
9436 Mirror Pond Dr.
Fairfax, Va. 22032

We can also be reached by phone at: (703) 503-3165.
The WPJ BBS can be reached at: (703) 503-3021.

The WPJ BBS is currently 2400 Baud (8N1). We'll be going to 14,400 in the near future, we hope.

- WPJ is available from the WINSDK WINADV, MSWIN32, BPASCAL and BCPPWIN forums on CompuServe, and the IBMPC, WINDOWS and BORLAND forums on GEnie. It is also available on America Online in the Programming library. On Internet, it's available on WSMR-SIMTEL20.ARMY.MIL and FTP.CICA.INDIANA.EDU. We upload it by the 1st of each month and it is usually available by the 3rd or 4th, depending on when the sysops receive it.

LEGAL STUFF

- Microsoft MS-DOS, Microsoft Windows, Windows NT, Windows for Workgroups, Windows for Pen Computing, Win32, and Win32S are registered trademarks of Microsoft Corporation.

- Turbo Pascal for Windows Turbo C++ for Windows, and Borland C++ for Windows are registered trademarks of Borland International.

- Other trademarks mentioned herein are the property of their respective owners.

- WordPerfect is a registered trademark of WordPerfect Corporation.

- The Windows Programmer's Journal takes no responsibility for the content of the text within this document. All text is the property and responsibility of the individual authors. The Windows Programmer's Journal is solely a vehicle for allowing articles to be collected and distributed in a common and easy to share form.

- No part of the Windows Programmer's Journal may be re-published or duplicated in part or whole, except in the complete and unmodified form of the Windows Programmer's Journal, without the express written permission of each individual author. The Windows Programmer's Journal may not be sold for profit without the express written permission of the Publishers, Peter Davis and Michael Wallace, and only then after they have obtained permission from the individual authors.

Table of Contents

Windows Programmer's Journal Staff:

Publishers	**Pete and Mike**
Editor-in-Chief	**Pete Davis**
Managing Editor	**Mike Wallace**
Contributing Editor	**David Campbell**
Contributing Editor	**Rodney Brown**
Contributing Editor	**Bill Lenson**
Contributing Writer	Jeff Perkel
Contributing Writer	Mike Strock

WPJ.INI

by Pete Davis

{Just a note, I drew the picture. It isn't very flattering. I'd like to think I'm not really that ugly. Pending a more appropriate picture, though, it'll stay.}

Ah, it's that time of month again (actually, a bit past it). As usual, we're late. Lately we've been getting VERY FEW SUBMISSIONS!!!!! Come on people, let's get things moving along. I'm serious, we're going to have to switch over to every other month if things don't pick up. We get a lot of mail from people saying, "yeah, I want to write about ..." and we say, "Great, let's see it." But it never shows up. I hate to start off an issue complaining, but really... Sure, everyone's busy, but surely you can take a few hours now and then to put together an article. Do you realize that if only 1% of our readers actually wrote an article we'd have more articles than we know what to do with? And still, 99.999% of you don't take the time to submit an article.

I know what you're thinking, what's in it for me? Well, let's start with my story. Sure, I'm one of the editors and the publishers, but this isn't exactly a 'real' magazine. Yet, I have signed a contract to write a book and I've got a two part article showing up in Dr. Dobb's Journal (more on this later).

Does that mean that any of you can do it? Actually, yes, it does. I'm no genius, I'm just a regular guy willing to take the time to do the work. You may remember Alex Federov, one of our earliest contributors. You can look forward to seeing his next article in Dr. Dobb's Journal also! Not to hack on Alex, he's a smart guy and a good guy. His english isn't perfect. Reading his writing, you know it's not his native language, yet here he is writing for Dr. Dobb's Journal only a few months after writing for us. I'm not saying that we're the reason he got published in Dr. Dobb's. But I doubt that writing his first few article for us did any damage.

You'd be surprised who reads this magazine. I only recently found out that Rob Burk, Editor of Windows/DOS Developer's Journal reads this magazine. I know it has been read by Andrew Schulman (see his letter in Issue #2). I am told that Mitchell Waite of the Waite Group Press has also read it. Whether or not these people read it regularly or not, I don't know. I'm not trying to pat myself on the back here, I'm just saying that articles written for our magazine don't necessarily go unnoticed by others in the industry.

I'm sorry to be going off like this, I should be chipper and happy, but I'd just REALLY HATE to see this magazine have to go to every other month. From the response I've gotten from a lot of our readers, they don't want to see that happen either, yet, where are their articles?

Ok, enough of that, you guys figure out if you want to get it every month. Remember, it's only going to take a few of you to make the difference and if you're counting on someone else to do it, forget it, they haven't done it yet.

On to happier things... Yes, as I mentioned, I have a two-part article coming out in Dr. Dobb's Journal in Andrew Schulman's "Undocumented Corner" column (which happens to be the same one Alex Federov's article is in). I don't want to say too much about it, basically because I haven't checked with Andrew about how much I can say. I will simply say it involves work I've been doing with the aforementioned Ron Burk of W/DDJ in the WinHelp area. There, that should get you wondering.

I've been REAL busy lately with this and the book, but it looks like I might be picking up some relief in the near future, again, can't discuss it now, but maybe next month.

Things are looking good for Windows NT. Looks like it's going to be shipping soon. If you are a Windows programmer and you have the hardware, I highly recommend getting it. If for no other reason, it's an incredible Windows development environment. I'm serious, it's NICE. Don't you hate it when you're testing a program, it crashes, and you have to re-boot? Kiss that problem good-bye. Under NT, your program crashes, and the session for that program closes. Nothing else is affected. Everything else works just fine.

Mike took a little flack for his 'review', last month, of an OS/2 2.1 review in a popular computer magazine. At the time I agreed with Mike, for the most part. You have to understand that what we write in WPJ.INI and the Last Page are not to be construed as fact or serious (well, sometimes: I'm dead serious about switching the magazine to every other month!). Mike's comments such as, "that useless operating system" is not meant to mean that OS/2 is useless, just an expression that he doesn't like it.

So, as I said, I agreed with him at the time. Recently I've had a chance to use OS/2 2.1 quite a bit. Do I still agree with Mike? Mostly, yes. I had to install OS/2 2.1 three times before it would actually let me switch between OS/2 boot and DOS boot, and back to OS/2. The first two installs it wiped out my \OS2\SYSTEM directory in the switch from DOS to OS/2. Now, after the third install (we're talking 18 disks and a real slow install program), it works, sort of. It still locks up quite a bit. It took me three re-boots the other day just to get past my network requester startup.

So, my opinions? I don't think OS/2 2.1 is particularly stable. I didn't think OS/2 2.0 was particularly stable either. I had many of the same problems with it. OS/2 2.1 is signficantly faster, especially for running Windows Apps, than it's predecessor, 2.0. Still, as I said, it's not very stable. I'll admit this was a beta, but let's compare it to another beta product, NT.

Windows Disassembler
Version 1.7 for Windows 3.1

* Disassembles Windows EXEs & DLLs
* Provides exported, imported, & API function names
* Previews code before generating files
* Detects data inside code segments & offers control over data/instruction translation
* Disassembles all 486 instructions
* Includes a utility that converts basic ASM files into MASM 6.0 high level sytax.

Only $20.00 per copy

Eric Grass
1612 Gettysburg Landing
St. Charles, MO 63303
(314) 928-7803

I've installed all three beta versions of NT. Each one, I installed once and only once. Each worked the first time. We did have a small problem with the October '92 version because of a hardware conflict which, once we posted a note on CompuServe, we received a fix within a few days. That's not a bad turnaround in my book. Try getting IBM to send you a "STABLE" version of OS/2...

As far as OS/2 2.0 and 2.1 performance, I think they are both slow, for the most part. When compared with Windows NT, there's no comparison. NT is blazing fast. I keep waiting to smell smoke coming out of the computer. When OS/2 runs, for one thing, just booting up takes forever. Then, when you finally boot up, it crawls, even with a machine that is well above the minimum requirements.

I haven't used NT with a machine at the bare minimum requirements, but I've used OS/2 on machines of equal caliber and there's no comparison.

Ok, so let me get my flack, but that's my PERSONAL opinion of OS/2 as I have experienced it. I know there are a lot of people out there that like it. That's fine, I'm not hacking on you. A lot of people like Snickers. I don't, but it's fine with me that they do. Understand? I think that was the way Mike wanted his article to be viewed. I think he's writing in response to one of the letters he got, so you can read that too. I think this is enough about OS/2 for a magazine that doesn't cover it!

On to other topics, that damn WinHelp file. Well, we've had some problems with it in the past, but I think we're going to be past them on this issue. I say that because I have been working with Ron Burk and in the process have learned a great deal about WinHelp of which I was previously unaware. We'll have the bitmaps in this time. We'll also have the browse buttons that we keep getting complaints on.

Our deepest thanks to Kerim Erden for a lot of the ideas and know-how to produce the WinHelp File. He came up with some really great ideas and showed us how to implement them. We're deeply indebted to him for the help. Hope everyone enjoys the new format. So, sit back, fire up Windows, and check out the latest issue.

Peace.

_Pete Davis

Letters

Date: 16-May93 11:49 EDT
From: Ian Pawson [100015,1364]
Subj: WinStub

Hi,

I was interested to read the article on enhancing the WinStub program on page 31 of the May 93 issue.

To make this a truly generic windows stub, it needs a couple of minor modifications to enable it to launch ANY program to which it is attatched. The spawnlp() function call needs to pass Windows the name of the .exe file so that Windows can run it after it has started. This is achieved by using the argc[0] var. (Note that argv[0] is the full dos path name, not just the program name.)

Two lines need to be changed and one added, as shown below:

```
#pragma argsused
int main( int argc, char *argv[] )  // get command line Parameters

returncode = spawnlp( P_WAIT, "win.com", " ", argv[0], NULL );  // pass
filename to Windows
```

The added #pragma command is just to stop the compiler complaining that argc is declared, but not used.

Cheers, Ian.

[Thanks for the note, Ian.   Good tip. - Ed.]

From: Jeff Miller <jmiller@hp1.terra.colostate.edu>
Date: Wed, 12 May 93 12:01:01 MDT

I enjoyed your Windows Programmer's Journal, v.1 n.5, but as much of a fan of Windows as I am, I feel that your "Last Page" column was misleading.

You mention reading an article about OS/2 2.1, and getting the impression that the author was trying to make OS/2 into Windows.   Of course OS/2 isn't Windows, but just like Windows NT, it is Windows compatible, which would make it of interest to any current Windows user.

The first thing that really got my attention was when you call OS/2 "that useless operating system".   What is your rationale for calling it "useless"? Have you run DOS, Windows or OS/2 apps under OS/2 (even version 2.0?) if you had, I hardly believe you'd think OS/2 was "useless".   Next you state that for Windows [under OS/2] you need "a 486 with 12MB of RAM and a 100MB hard drive".   First off, the minimum requirements of OS/2 (at least v2.0) are: 60MB hard drive, 4MB RAM, 386SX+ processor.   Of course this won't be an optimal setup, but I would consider this setup a minimal (practical) Windows 3.1 setup.   The last I heard, Windows NT will require a 386, 16MB RAM (that might have been for the beta - perhaps it's 8MB? [According to the Microsoft press releases, the production version of NT will require 8MB. The beta NT "requires" 16MB if you include everything, but this requirement can drop to 8MB if you remove the networking software. -Ed.]) and 80MB (again for the beta, perhaps less on release) for the OS.

You mention that you could run OS/2, but you've opted for Windows NT instead. How much space is that operating system taking? How much RAM do you have? What type of processor do you have (and what speed)? Have you tried comparing the speed of Windows apps under OS/2 vs. Windows (3.1) apps under Win NT? From the beta versions of Win NT I've seen, I'd have to believe that OS/2 would be faster.

In closing you say "If you want to run Windows, why buy OS/2?". I would say because OS/2 (and Win NT) will give you more performance and features out of that machine that's currently running a DOS based GUI extender. Hey, just try formatting a floppy under Win 3.1 and doing anything else. You go on to say "anyone buying OS/2 on this promise [to run windows apps better than Windows] will only be disappointed, I think." Why? OS/2 (v2.1) seems to run Windows 3.1 apps just as if they were under Windows 3.1. What's disappointing about that? Lastly you say that "an operating system should be able to stand on what makes it unique, or it will end up being nothing more than a cheap imitation". I agree, but this doesn't apply to OS/2. You make it sound like all OS/2 does is run Windows 3.1 and DOS apps with a different look to it. OS/2 as an operating system is very powerful, even without the Win3.1 and DOS compatibility. Remember that compatibility doesn't necessarily mean there's no innovation involved.

Well, I hope I haven't come off sounding too harsh. I _am_ a Windows 3.1 user, but I felt in fairness to OS/2 that unsubstantiated claims shouldn't go unchallenged.

Jeff Miller

[Thanks for the letter, Jeff. I was getting the idea no one was reading the "Last Page". I'd rather get negative mail than no mail at all. You had some questions/comments I want to address, so here goes:

a) OK, perhaps "useless" was too strong a term. My experience with OS/2 has been limited, but I haven't seen anything to make it stand apart from Windows (particularly NT). My main point of contention is stability - I've heard several horror stories about OS/2 2.1 locking up regularly (see Pete's "WPJ.INI" column in this month's issue). Under NT, if a program crashes, only that program stops running - nothing else is affected. The only problem I've had with NT was the NTDETECT.COM file was incompatible with my monitor. A note posted on CompuServe netted a new NTDETECT.COM file within a couple of days and the bug was gone. If you haven't had problems with OS/2, then I'm glad to hear it. As a programmer, nothing is more frustrating to me than an OS that hinders my productivity. I only used the word "useless" based on my own experience with OS/2. If there are advantages to running OS/2 instead of NT, then I haven't heard them.

b) I'm running Windows NT on a 486/50 with 16MB RAM, and it takes up about 72MB (including the 24MB swap file) on the hard drive. Again, my experience with OS/2 has been different than yours. I've seen OS/2 on more powerful PCs than the minimum required for OS/2, and it crawls compared to NT. When I wrote about the minimum requirements for OS/2 (e.g., 16MB RAM), I was referring to a practical setup. I apologize for the misunderstanding. I've seen OS/2 run on a PC with a setup less than the one I described as practical, and, to me, it was just unusable when it came to response time. If your experience differs, then that's great. I hope other people have had better luck with OS/2 than I have.

c) I will be the first to admit that Windows 3.1 has its limitations. Anything based on an OS as old as DOS will have some problems, especially if you're developing software. Your point about formatting a floppy under 3.1 is a valid one. This is why I have NT. It avoids DOS problems such as the 64K limit on segments. Sure, so does OS/2. But what does it offer that NT doesn't?   You write that OS/2 "...is very powerful, even without the Win3.1 and DOS compatibility."   If you want to elaborate on this, we'll be glad to print it.   This brings up another issue for programmers: availability of programming aids.   When I go to the bookstore for help on programming Windows, there is a plethora of choices for me to look at and decide which ones will help me the most.   I've looked for books on programming OS/2, and I've rarely seen any.   Since you read WPJ, Jeff, I'll assume you're a programmer.   How many book have you seen on writing OS/2 apps?   I hope you've had better luck than me. One measure of the success of an operating system should be the the amount of literature available on programming in it.   This is another reason I choose Windows (3.1 and NT) over OS/2.

I don't think your letter was too harsh.   On the contrary, I'm glad you took the time to write. I'm not trying to preach the gospel, just write about my own experiences.   If people are happy with OS/2 and they're productive, then more power to them.   OS/2 just isn't for me. If you (or anyone else) has comments about my response, then please write.   I hope I've been fair to Jeff in my response to his letter.   If you disagree, let me know.   Thanks for the letter, Jeff. -Mike]

Hi,

I'm a novice windows programmer, and I just wanted to write to you to thank you, and encourage you to continue your efforts, and those of your other writers, in the Windows Programmer's Journal.   It is quite useful for   people like me, with columns for both beginners and advanced programmers.   I'm especially glad that you are expanding the authorship to include other topics.   Keep up the good work!

Thanks

jeff
perkel_j@a1.mscf.upenn.edu
71404,123

[Thanks Jeff.   I'm glad you like the magazine so much.   All past and current authors should give themselves a good pat on the back.   You've all done a great job in helping us put out this magazine every month.   We couldn't have done it without you.   In this month's issue, you may notice an article from Jeff.   Thanks for helping us out with the article, Jeff. Now, if only we could get more people to show us their appreciation for WPJ the way you have. -Editor]

Beginner's Column

By Dave Campbell

Ok, I'm gonna be honest here. It is June 6th, Pete and Mike have been very understanding, and I'm late. I've got a diet coke, and a bag of cheap donuts, and it's time to go. Don't try this at home boys and girls, it should only be done by .... (guess that leaves me out).

I promised to put the file dialog from last time into a DLL, and I am not going to make it. Instead I am going to answer a couple of people's questions about Borland vs Microsoft in terms of make files.   The files as I have been submitting them so far have been Microsoft compatible (but you Borland turkeys already knew that, didn't you?). I am in a sort of enviable position (except for disk space) in that I have both compilers at-hand. I started out Borland, and will still go that way for expedience, but I am trying to become more literate of Microsoft also.

In this month's HELLO.ZIP file, you will find two make files. The two are named HELLO.MAK (for Borland), and MHELLO.MAK (for Microsoft). This is how I will ship from now on. The syntax for executing the make is:

1) Borland
```
make -fhello.mak
```

2) Microsoft
```
nmake -fmhello.mak
```

The Microsoft make file is void of pathspecs and such compared to the Borland make file. The reason for this is that the Microsoft compiler builds all that stuff into your environment when it installs, so my environment contains (amongst other things):

```
TOOLROOTDIR=E:\MSVC
INCLUDE=E:\MSVC\INCLUDE;E:\MSVC\MFC\INCLUDE;
LIB=E:\MSVC\LIB;E:\MSVC\MFC\LIB;
INIT=E:\MSVC;
```

This allows the make file to be somewhat more streamlined.

Building Make Files

If there's one piece of code that could use an entire book written just for it, it is the make files. I don't care if you are Borland or Microsoft-compatible, they are both confusing. So, how do I build them? Simple, I steal them. Honest! (or is that Dishonest!?)

My first choice of a make file was the last one that worked. If that isn't possible, and I am using Borland's tools, I get into the IDE, select Project/Open, give it a name, and then use the Ins key to add the filename.c and filename.rc. Then go to Options, and setup a small model Windows EXE file default selection, and save everything. I get out of the IDE, and run:

```
PRJ2MAKE <filename> <filename>
```

This converts the <filename>.prj file to <filename>.mak that we wanted. Once I have a make file, I can modify it like a madman, but just getting the first one drives me crazy.

This same technique can be used (sort of) with VC++. The difference is that part way through the process, it asks if this is to be an external make file, and you say YES.

Hello

I have made some slight changes to hello this time around, besides the two make files. I added the items necessary to produce 3.0/3.1 files with newer tools:

```
#define WINVER 0x0300
```

This line must be before #include'ing windows.h, and:

```
RC -30 hello.res hello.exe
```

This is in the make file, and instructs the RC executable that we want to execute under both 3.0 and 3.1.

The only other change is the line:

```
wc.lpfnWndProc = (WNDPROC)WndProc;
```

in the InitInstance function. This fully qualifies the WndProc pointer by casting it correctly.

Tease

I must apologise for a short article, but as you can see, I am late, and have been jammed all month. Next month we'll continue with hello.c and the DLL carry-over from last month.

In addition, next month I'll explain the code I put into hello this month. I quickly coded up a surprise that will lead into a short discussion next issue, and a series of follow-up articles in addition to the Beginner's Column. I'm not going to discuss it here, you have to find it, and execute the code to see what I've done.

Have fun, and enjoy the code!!

EndDialog

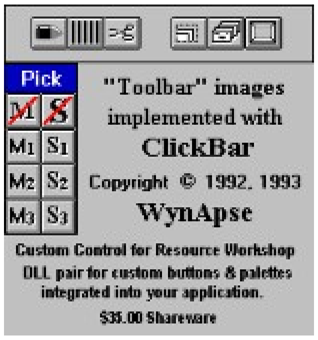
Another month down the tubes, so to speak, but before I go I do want to mention one thing. It is my opinion that any code I put in this magazine is in the public domain as of the time Pete and Mike post it. I pull code from books and magazines, and anywhere else I can find it to save me time and get a job out quicker, and I am not about to put any restrictions on anything I talk about here. That's all.

Next month, we do the DLL, so watch for us. Feel free to contact me in any of the ways below.   Thanks to those of you that have e-mailed me questions, keep them coming. Notice that I now have an internet address, to make it real easy to get in touch with me. I want to rat out the things other people are having questions about, not just what I think people want to hear.

Dave Campbell

WynApse PO Box 86247 Phoenix, AZ 85080-6247 (602)863-0411
wynapse@indirect.com
CIS: 72251,445
Phoenix ACM BBS (602) 970-0474 - WynApse SoftWare forum

ClickBar consists of a Dynamic Dialog pair that allows C developers for Windows 3.0/3.1 to insert a "toolbar" into their application. Microsoft, Borland, etc. developers may display a toolbar or tool palette inside their application, according to a DLG script in their .RC or .DLG file.



"Toolbar" images implemented with ClickBar
Copyright © 1992, 1993
WynApse
Custom Control for Resource Workshop
DLL pair for custom buttons & palettes integrated into your application.
$35.00 Shareware

Borland developers may install ClickBar as a custom control, providing three custom widgets added to the Resource Workshop palette. These are three different button profiles defining 69 custom button images total. The buttons may be placed and assigned attributes identically to the intrinsic Resource Workshop widgets.

Source is provided for a complete example of using the tools, including the the use of custom (owner-drawn) bitmaps for buttons.

Version 1.5 uses single-image bitmaps to reduce the DLL size, and includessource for a subclass example for inserting a toolbar.

Registration is $35 and includes a registration number and printed manual.

WynApse
PO Box 86247 (602) 863-0411
Phoenix, AZ 85050-6247 CIS: 72251,445

Internally Yours

By Pete Davis

It's about time we had another book review, and every once in a while a book comes by that I can't wait to review. ***Windows Internals*** by Matt Pietrek is just such a book. This book is published by Addison-Wesley and is the latest in the Andrew Schulman Programming Series. You might think I'm a little biased since my book will be in the same series. I can't exactly deny that, but I try to be objective when doing book reviews.

I have been waiting a long time for this book to show up and I have to say that I'm not disappointed at all. ***Windows Internals*** provides a view at the inner workings of Windows that you won't find anywhere else, unless you work for Microsoft and have the source code., of course. To write the book Mr. Pietrek reverse-engineered many different parts of Windows. From the disassembly he has managed to create C-like pseudo code to show how the different routines inside Windows work.

One of the things Mr. Pietrek discusses early on is how easy it is for one to forget that WIndows is nothing more than a program or, more accurately, a collection of programs. There's no magic involved. When it comes down to it, the different parts of Windows are simply routines that perform operations, just like any other program.

The main purpose of this book is to provide an understanding of how things work under the hood, as opposed to providing some sort of direct programming aid. The idea is that if you understand how things work internally, you'll understand why a function works the way it does.

Chapter 1 is entitled "The Big Bang" and covers the startup and shutdown processes of Windows.   From WIN.COM to exiting Kernel and dropping back into DOS(without all the stuff in-between, actually), you see what Windows is doing to create its environment. There's some really good information in here that lets you clearly see how Windows is really a DOS extender with a GUI wrapping. All of the pseudo code is clear and well documented.

Chapter 2, "Windows Memory Management", will have a bit of a broader appeal to most of you. Chapter 2 begins with a description, in brutal detail, how the selector functions work. As selectors are the basis for the Local and Global memory mangement routines, it provides a strong foundation for the information which follows, the Local and Global memory management functions, of course. I believe there is pseudo code for EVERY Global and Local function, including many of the lower-level functions (unavailable to the Windows programmer) used by the Global and Local functions. If Mr. Pietrek missed any, you'd be hard-pressed to find them. This is the information that I found most useful. It really gives you a good feel for Windows as a DOS extender.

Chapter 3, "Starting a Process: Modules and Tasks", again, goes into gory   detail, this time on how programs are loaded and run by Windows. In addition, he also covers the loading and execution of Win32s programs under Windows 3.1. Another very thorough and informative set of functions in pseudo-code.

Chapter 4, "The Windowing System", is what I considered the second most important chapter. You need to keep in mind that everything on the screen in Windows is a Window (just about). A button is a window, as is a scroll bar or a dialog box. Keeping that in mind, the chapter explains how the windows are kept in memory, how they are displayed, etc. Many useful functions are pseudo-coded.

I must admit I haven't given chapter 5, "The Graphics Device Driver Interface", the attention it deserves, yet. It covers selected GDI functions such as CreateDC, CreateBrush, etc. There enough of it to keep you busy, if that's your cup of tea.

Chapter 6, "The Windows Scheduler", is another favorite of mine. This has some fantastic information on how tasks are scheduled and prioritized. I personally have quite an interest in task management, so I really spent some time here. I, personally, would have preferred a little more in this chapter, but that is a personal preference.

Chapter 7, "The Windows Messaging System", is another important chapter. Messages are the heart of Windows programs. Understanding how messages and message queues work at a lower level gives a lot of real insight into how Windows programs will react in different situations.

The last chapter, "Dynamic Linking" is a bit shorter than I would have expected, but in retrospect, I can't think what I would have added. Again, painful detail is the name of the game.

Ok, I admit it, I liked this book a lot. I know it might be hard to see me as objective, when reviewing a book in this series, but I think that under any circumstances I would feel the same about this book. It's well organized, extremely detailed, and covers most of the major areas you'd expect to be covered. There is a hint of a second book on Windows Internals. I see this as a fix for the one problem the book has, it doesn't have everything. It would probably take more than two books to cover ALL of Windows, but two should cover just about all of the most important areas.

I recommend this book highly to anyone doing Windows programming. This book gets into the nitty-gritty details of implementation. It's not light or easy reading, but it provides a solid understanding of how Windows works under the hood. Any programmer who's had to wonder why Windows did something in one circumstance and not in another will probably find either the answer or some good clues in this book. Again, I recommend it highly. Go to your local book store and check out a few chapters. If you can understand it, then you'll want to get it, trust me.

Pascal in 21 Steps
(A Beginner's Column for Turbo Pascal for Windows)

By Bill Lenson

Step One - Fundamentals
(or, "you can't get by without this stuff")

OK, here we go.   Last month we talked about what to expect from this Pascal column.   In brief, this is going to be a crash course designed to introduce any devoted reader to the fundamentals of programming Borland Pascal for Windows.   At the end of this months column you should be able to write a simple program to display a calculation to the screen, know what a program is, what a variable is, and what some common data types are.

Try not to get too hung up on one or two concepts you may not understand the first time. Read this through once and follow the instructions.   If it's not clear, read it through again until you master the problem. Sometimes understanding just 'how' to do something is better initially and then the 'why' gets solved by itself later.   This first column may seem a little wordy but I think it needs to be to make sure everyone understands the basics.   Future columns will progress much quicker.

If you have any ideas for future directions with this column, or any comments and have access to e-mail on a Bulletin Board, please send them to me at the address at the end of this article.   Thanks in advance!

Where to begin?   Ah yes, the beginning.

1) A Very Brief History of Pascal

Pascal is one of several hundred "languages" you use to talk to a computer.   Originally it was designed as a learning tool by a Swiss Professor of Computer Science by the name of Nicklaus Wirth.   It seemed that other languages had too many barriers for his students to easily learn fundamental computer concepts.   These computer concepts are called Data Structures and Algorithms.   We'll talk more about what these are in a later column.   As an aside, one of Nicklaus' early students later went on to write his own adaptation of Pascal. That person is Philippe Kahn, founder and CEO of Borland Corporation, and his Pascal extension is, of course, Turbo Pascal.

OK, we know Pascal is a language which we use to talk to the computer.   Why not just talk to the computer in English.   Frankly, it's too difficult to do efficient "natural language" processing (although this is one of the hottest topics in Universities today).   English has thousands of words with multiple meanings for each so we settle for Pascal with it's fifty or so common words in it's language.   How you compose a program is the topic you will be studying from now on.

2) Programs

The Pascal language is concise.   We only need to type one or two words to get our point across and instruct the computer to carry out our wishes.   For instance we can write some text on the screen or read numbers we type from the keyboard.   But what is a program and how does the computer understand it?

Programs are just a series of instructions written out in a text file.   Borland Pascal comes with a text file editor which you should use to type in your programs.   After typing them in you can save them to disk just like a letter in a word processor. The Borland manuals and online help will give you plenty of help with the editor and how to use it.

Now, you have a program, so how does the computer know how to execute the instructions written out in that file? It doesn't yet because (don't worry I'll explain this) computers don't understand Pascal. Computers only understand one language known as machine language. Machine language is very powerful but very difficult to learn. What you can say with one instruction in Pascal could take 100 instructions in machine language. Now we're still left with a problem: we write programs in the Pascal language but the computer only understands machine language. What do we do? And the answer is - Borland Pascal comes with a translator which will convert Pascal to machine language. It will even tell you if your program isn't written properly (at least it will find spelling mistakes). This magical translator is called a compiler. And what a compiler! Borland Pascal's compiler will translate up to 85,000 lines of Pascal code to machine language in one minute.

The Borland Environment is complete. You can compose your programs using the editor, compile them to machine language using the compiler, and it will let you execute the program and tell you where problems are. The compiler will even make an EXE file so you can run it at any time. There's even a debugger built in. A debugger is a program used to help you find potential problems in your program. Here's goes another tangent... The term bug actually dates back to the 1950's when an early computer suddenly stopped executing a small program it was running. When the researchers opened up the back and looked at the mounds of wires and vacuum tubes, they discovered a moth had flown in and shorted it out. From this time forward, whenever a program "crashes" or does something unexpected, it is said to have a bug in it somewhere.

3) The Underlying Idea Behind Programming

The idea behind writing a program is relatively simple. You aren't really programming a computer, you're controlling the devices and memory that make up a computer. Whether you program for DOS or Windows, the idea is the same. All too often, programmers forget this and find themselves writing code that is just plain crazy.

So what are the devices? Well, there are input devices which supply data to your computer and output devices which do something with data coming from your computer. Examples of input devices are: keyboard and mouse. Examples of output devices are: display and printer.

Memory is a little different. I like to group memory into two categories. First there is RAM which acts like a temporary storage for numbers like a memory on a calculator. The difference is that calculators usually only store a few numbers, RAM can store thousands if not millions of numbers. Like calculators, RAM gets erased when you turn your machine off. The other type of memory is a more permanent store, the disk. This can be a floppy diskette or a hard disk. You can write to and read from disks just like you can from RAM but data on disks are persistent (doesn't go away when you turn your computer off, only when you explicitly erase it). For sake of clarity, I'll call "RAM" memory and "disks" disks.

Behind all this is the controller, the thing you talk to to get things done, the thing that processes all your commands, the Central Processor Unit (CPU). The CPU reads instructions one after another and carries out what they're supposed to do. Some instructions read input, some write output, some read or write from the disk drive, and some manipulate data we store in the RAM. What's important is programs are usually driven by taking input from the user and storing the information in RAM, performing some calculations on the data, and either store the results on a diskette for later use or display the results on an output device.

This can be illustrated by the old-faithful computer drawing:

```
                                +---------- +
                                 |    RAM    |
                                 |                   |
                                +-----------+
                                       /\
                                       ||
                                       \/
+-----------+          +-----------+           +-------------+
|   Input       |----->|    CPU      |----->|    Output    |
| Device |             |                   |            |    Device   |
+-----------+          +-----------+           +-------------+
                                       /\
                                       ||
                                       \/
                                 +-----------+
                                 |   Disk        |
                                 |   Drive      |
                                 +-----------+
```

Fig. 1) Typical Flow of Data in a Computer

It may not become apparent how important this drawing is right now but when you get better at programming you'll realize that each block is a major area of study. Input and output device control should be fairly obvious - you want the most efficient and user friendly inputs and the clearest, most user friendly output. RAM study focuses on Data Structures which are different ways to think of data stored in the RAM while running your program. Disk drive study is the study of database design. No matter if you're saving a mailing list, word processor letters, or EXEcutable files, a disk drive can be thought of as a filing cabinet. How you organize those files is the real trick. When you can successfully control each of these and have your program perform something of value, then you've learned to program.

OK, enough of sounding like Master Po from Kung Fu. Let's get down to the real stuff - writing a program.

4) Writing Your First Program - Hello world.

So, now you're armed with the compiler and editor waiting to blast away at the first program. Good! I'm assuming you are using Borland Pascal for Windows and you have it loaded up on the screen. The first thing you'll need to do is change some environment Options to make sure what I tell you will work.

Click on Options, then Environment, then Preferences. Click on the field that says Alternate in the top right corner. This ensures the keyboard presses I do will function the same for you (for example, F3 will load a program and F2 will save it to disk now). Click OK. Click Options, Environment, then Editor and change Tab Size to be 4. I believe TAB will be set to 8 spaces by default. Click OK. Click the Options menu option one more time and then Click on Save. You're now set up with the same basic editor options as me.

To write your first program press F3. A file open dialog box should appear. Type EX1 and press Enter. You have just created a blank file called EX1.PAS which is stored on the disk. The editor is now opened up and waiting for you to type in your program. Type the following EXACTLY!

```
program HELLO_WORLD;
uses WINCRT;
begin
    writeln('Hello, World');
end.
```

When this is entered, press F2 to save it to disk. Now hold down the Ctrl key and then press F9. If you typed everything correctly you should see a window appear on the screen with the words Hello, World written in the top left corner. If you didn't get everything quite right, you will probably get an Undefined Identifier message which just means you spelled something wrong. The editor will even highlight the line that has a problem. Compare your program with mine in that case. To close the window, double-click the mouse on the close box in the top left corner of the window you just created.

Your first program! Well done!

OK, it's not much but it's a start. But what does it mean? Funny you should ask... There are several things I should point out about this program before we go on.

Every program is made up of a series of statements. Statements end in a semi-colon (;) and are kind of like a sentence in English. The compiler will translate each statement one by one. In the program above there are three statements. Begin and End act like book-ends and surround instructions of actions to carry out. In this program, there is only one instruction (writeln('Hello, World');). The last End in a program always has a period after it. Other Ends, if present, usually have a semi-colon instead.

The first line of a program is really just for information purposes and gives a one word description of what the program does. Notice I said one word. That's why it's spelled HELLO_WORLD, not HELLO WORLD. The underline tricks the compiler into treating it as one word. Another term you might hear every now and then is identifier. An identifier is often used to describe words you make up and place in programs. HELLO_WORLD is something we made up. We could just as easily have put THE_FIRST_ONE, or EXAMPLE1. Program is part of the Pascal language so we usually don't call it an "identifier". We call words that are part of the language "reserved words".

The second statement, USES WINCRT;, well, let's leave that one for later.

As we said above, the Begin and End surround a group of instructions. In the future we'll have many more instructions inserted where our one lonely writeln instruction (pronounced "write line") is now.

Our first program deals with two of the blocks in the diagram above: CPU and an Output device (Window on the screen). The CPU was used to execute the instruction. Now we'll turn our attention to memory and talk a little bit about variables.

5) The Second Program - Hello World 2.

Any program of significance is made up of two parts, instructions and memory storage. The number of instructions that manipulate memory far outnumber the number that control devices and disks.   We better talk about memory and variables before we go too far.

Machine language programmers often think of memory as a long string of millions of bytes (MegaBytes).   Pascal treats memory more abstractly as small chunks.   Each chunk can store different types of data such as numbers or strings of characters (letters, numbers, symbols).   In Pascal, we call the chunk a variable.   We don't care where in memory a variable is located or how big it is.   We tell the compiler we want a variable that will hold a number 4 digits wide (for example), and it sets aside a section of memory for our use that is big enough for that number.   So we can refer to variables in our program, we give them names. Some examples of reserving variables are:

```
x      : integer;
ch     : char;
name : string;
```

These are three variable declarations telling the compiler to reserve three distinct chunks of memory.   The first variable, x, will hold integer numbers. We say that x is a variable which can store a data type of integer.   The next one, ch, is a character type variable and will hold a single character (letter, number (0 to 9), or symbol).   The last variable is a string var (variable) called name which can hold up to 255 characters.   From its name, 'name', we can probably conclude it will be used for storing somebody's name for later processing.

Integer, String, and char are three of Pascal's pre-defined data types.   There are others, which we'll study in time, that reserve chunks of memory too.

How do we put variables into a program and where?   Pascal needs to know about variables and their types before the instructions can execute them. We therefore declare them to the compiler, before the begin, in a 'var' section.   Here's an example to illustrate variables better.   To type this in, press F3, type EX2 and press the Enter key.

```
program HELLO_WORLD_2;
uses WINCRT;
var
    s : string;
begin
    s := 'Hello, World';
    writeln(s);
    s := 'From Bill';
    writeln(s);
end.
```

Don't forget to press F2 when done typing this program to save it to disk.

In the above, var starts a series of variable declarations.   Another way you can think of a variable is like a box.   You can put one thing in the box and then take it out but only one thing can be in the box at one time. Also, the box can contain only one kind of data.   From the program above, we see a variable called s that can hold a string of up to 255 characters.

The first instruction is called an assignment statement.   ':=' (pronounced "assign") means take what's to the right and move it into the variable on the left.   I could have more easily written the instructions as:

```
begin
    writeln('Hello, World');
    writeln('From Bill');
end.
```

but this seemed like more fun and I could explain variables along the way. So, the first instruction moves the string into the string variable. The second instruction writes the string to the screen and then moves the cursor to the beginning of the next line. s then gets assigned a different value, 'From Bill', erasing the previous contents. It too gets displayed in the window. Before you run this program, what do you think the output will be? Try the program and see if you were right.

6) Conclusion

There you have it. You should be able to now look at sample programs and at least spot the variables and the main Begin...End statements at the bottom of the program file. The function of the programs might be a little advanced yet but this will soon change.

Next month things really start to fly. You'll learn more about variables, data types, functions, procedures, and units.

See you next month!

7) How to Contact Me

Please don't try to contact me for programming problems. I have a full time job and won't have time to respond. If, however, you would like to send me comments or suggestions for future columns, I can be reached the following ways:

a) From CompuServe

- Type GO MAIL
- Enter your message
- When done and prompted for the mailing address put:

    >INTERNET:Bill.Lenson%canrem@uunet.ca

- The address must be entered EXACTLY as shown. Any character left off will cause the message to be sent but won't get to me.

b) From a Bulletin Board with Internet Access

A surprising number of BBS's have Internet access. If you have a local mail system, there's a good chance it could also have a "gateway" to Internet.

- Contact your BBS Sysop or mail administrator to find out how to send mail through Internet.
- Address your message to:

    Bill.Lenson%canrem@uunet.ca

c) From a BBS with FidoNet Access

- Contact your BBS Sysop to find out how to send mail through FidoNet.
- Address your message to:

Bill.Lenson@f15.n229.z1.fidonet.org

Please note that in the three addresses given above, there are NO spaces embedded anywhere in them. It's one character after another.

There are several mail systems which can send mail through Internet. If you are on any of the following systems you MAY have access: AppleLink, AT&T Mail, Bitnet, BIX, BMUG, CompuServe, Connect, Easynet, Envoy, FidoNet, GeoNet, MCIMail, MFENet, NasaMail, PeaceNet, SIGNet(through FidoNet), SINET, SPAN, SprintMail, THENET, or UNINet. Contact your mail administrator to see if you can mail through Internet. If so, drop a message and your comments will be considered for future columns.

Moving Away from ***Moving into Windows NT Programming***

By Pete Davis

Boy, I'm really going to sound biased this month. I can see the flames from here. In my review of ***Windows Internals*** I had nothing but good things to say. This review is of the book ***Moving into Windows NT Programming*** published by SAMS Publishing and written by Jason Loveman.

First of all, I must give Mr. Loveman some credit. His book is the first real Windows NT programming book to come out (that I've seen), so he really had his work cut out for him. Also, the book was finished only weeks after the third beta release of Windows NT, so the book can sort of be considered a "beta" book. As soon as I started looking through it, there was one thing that stuck out really bad to me and left a real bad taste in my mouth. About 1/6 of the book is devoted to a chapter called "A Port of WINIO." WINIO, for those of you not familiar with it, is an API to allow programmers to perform console-type operations (printf and the like) under Windows. In addition it allows you to structure your programs more like DOS or Unix C programs with a main() function. Well, that sounds good, so why would that leave a bad taste in my mouth? The authors of WINIO are David Maxey and Andrew Schulman, for one. Not Jason Loveman. Again, I tried to be objective, though, and bought the book anyway. I started reading that one chapter, just to see if I could find a justification, not for having it, but devoting 1/6 of the book to it.

After reading a few paragraphs here and there, it quickly became clear that the only true purpose WINIO served was to give the author the pages he needed to finish the book. To quote the author, "Changing all the code to bring up a simple WINIO application ... took about 20 minutes ..." I'm sorry, did you say 55 pages (the number of pages required for the dump of the WINIO code) took you as long as 20 whole minutes? "Actually, there wasn't much work at all.", he says. Well, really, and that was worth 55 pages? I'm sorry folks, but I paid $40.00 for this book and I'm saying, if it's not much work to do this **55 page** sample port, perhaps it wasn't the best example. In fact, he says that most programs would have a little more work involved. It seems to me that maybe one of these 'other' program would have been a better example. Ok, but enough of that. Let's look at the book and pretend that chapter doesn't exist.

The main focus of the book is porting existing Windows 3.x code to the Win32 API. It also stresses portable replacement of code as opposed to simply modifying the code to be Win32 compatible. These are good goals and the author makes it clear that these are goals. The 1st, 3rd, and 4th chapters do a reasonably good job of this, though the 4th is simply the WINIO port. Oops, I was going to forget about that, wasn't I.

Chapter 5 is titled, "NT Multitasking". Again, I was a little disappointed by this chapter. The author skims over the sections on Threads and Synchronization. I know, I'm kinda big on these topics, but I feel they are very important topics and they're probably going to be the two toughest problems for most Windows 3.x programmers to overcome when going to NT, at least, that's my opinion. Unless you've used threads before, there's a lot to learn. I think it is unreasonable to assume the reader of this book is going to have a working knowledge of threads and synchronization. There also doesn't appear to be any real organization to the chapter. First he talks about Objects, then DLLs, then Process synchronization, then on to InterProcess Communication (DDEML in particular), and then, whoah, Asynchronous Objects where he covers more about synchronization objects. It seems to be a sort of mish-mash of different topics without any sort of cohesive theme.

Chapter 6 is titled, "The NT is for New Technology". It's more like "the NT means I have 'No Title' better than this". The title gives no insight into the contents which, again, is a bunch of unrelated topics that could be either here or in chapter 5 and you wouldn't notice the difference.

Chapter 7 is titled, simply "Win32s". Believe it or not, I actually enjoyed this chapter. There was some really good information about Win32s and how it relates to Win32. In particular, he covers areas where they are different. This is useful information and it is well presented.

Chapter 8, "Portability" started off well. The first 8 pages discussed important portability questions. This is then followed by a sample program using the MFC libraries. The purpose, being to demonstrate MFC. I have nothing against MFC, but, again, it just sort of sticks out after the first 8 pages and seems oddly placed.

Chapter 9, "Tools" is a fairly mediocre discussion of the compiler options, linker, etc. Nothing really earth shaking.

Ok, I'm sure I'm going to get some serious heat for being so biased on this review. Different publishing house, doing a book on the same topic that my book will be on (sort of, not exactly). Really, when some more books come out, I can almost guarantee that I'll bring some up that are really good books and I'll be objective about them. I've tried to be objective about this book. It's hard to feel objective when you feel like you got burned, though.

The overall appearance of the book shows signs of average writing. There are a lot of editing errors that also stick out. There are sentences that I thought didn't make sense the first time and never made sense after that either. The organization of the book doesn't seem to be very well thought-out. There appear to be a lot of sections of unrelated topics thrown in together. The book's initial goals seem to sort of get lost after the first few chapters.

The book does have its good points. It does have some good information on porting existing code from Windows 3.x to Windows NT. The problem is, I've seen an equal job done in Microsoft's documentation. There's not $40.00 worth of new porting information.

I cannot recommend this book. I will admit my expectations have been a little high for the first Window NT/Win32 programming books. I wanted to see some good writing. I like a good programming book as much as the next nerd. Unfortunately, I don't think this is such a book. You can certainly check it out at your local bookstore and see what you think. Skim through it and see if you agree or disagree. Personally, I'd like to hear from readers who have already paid for the book to see if maybe they disagree with my review. This is the first negative review I've done, and I feel a bit guilty, but I also feel there's a need to warn the readers. Again, if there are readers out there who disagree with my review, I'd like to hear your responses.

Putting together
C++

C++ Beginner's Column

By Rodney Brown

Welcome to C++ For Beginners. Like Mike, I am also just starting out in C++, but I can see the many advantages, and some disadvantages that C++ has. For starters, I am going to devote this column to each of the new features that C++ provides. Once we get these basics down pat, we will then start on a (No, not another one!) 'Hello World' program (You knew it was coming, didn't you?).

Some of the items we will be discussing in the next few months are:

Objects/Encapsulation
Polymorphism
Inheritance
And much much more

I'm sure you will enjoy learning C++ as much as I am. Next month, we will begin looking at the differences between C and C++, and begin discussing Objects.

Due to time constraints (my hard drive crashed!), I was not able to write a complete article. If you have any suggestions, obstacles you can't get around or hints/tips, please send them to the e-mail addresses shown below. Next month, we'll have a full-fledged article for you. Sorry about that.

My E-mail addresses are:

CompuServe: 72163,2165
America Online: RodneyB172

Since my last article, I dropped GEnie for America Online

You can also contact me at the following Internet addresses:

CompuServe: 72163.2165@compuserve.com
America Online: RodneyB172@aol.com (Preferred InterNet Address)

See ya next month!

Windows Programmer's Journal BBS

By Pete Davis

I'm going to start doing updates every few months on what's going on with the Windows Programmer's Journal BBS. I thought that, apart from informing you, the reader, about what's going on there, it might provoke some of you to tell us exactly what you'd like to see there.

The Windows Programmer's Journal BBS, right now, is still in its infancy. I haven't had the kind of time I'd like to really get it configured the way I want. One of my main objectives is to get the WPJ BBS on FidoNet. Configuration for FidoNet is no small affair and requires a good deal of work. I have some vacation time coming in July and I'm hoping I can get to that then. Once on FidoNet we'll be able to open ourselves up to some of the active conferences out there and, hopefully, maybe even get one started for the magazine eventually.

Right now the BBS only has 128 megs. Much of this is taken up with software that isn't really necessary for the BBS. I'll be removing some of that soon as I get a new computer to relocate the software to. In addition, I plan on upgrading the modem to 9600 baud. This will make it easier on many of our long-distance callers. Both of these should be in place within the next 4-6 weeks. We also have a CD-ROM drive which currently houses the SIMTEL CD-ROM. The CD contains more than 600 megs of public domain and shareware software. We currently have two other CDs that we hope to make available to the BBS as soon as we can get additional drives. The other major purchase for the BBS is a 1+ gig hard drive. This purchase will be put off for at least the next 6 months due to a shortage of green paper.

The biggest news on the WPJ BBS front is the APIs in the download section. I've been working hard to get APIs and have found some pretty good ones and more are coming soon. Among the ones there now are: OLE Specs for 16-bit and 32-bit Windows, ODBC Specs, Message API Specs (MAPI), and much more. Soon we'll be adding the Telephony API and a few other odd ones that have reared their heads. It seems Microsoft is going API crazy. I'm a bit of an API fan myself, so no complaints from me.

As I said, I plan on having some time early in July to get some work on the BBS done. Please leave us comments and suggestions to let us know what you'd like to see.

The Windows Messaging System

By Mike Wallace

An integral part of learning how to program in Windows is understanding the Windows messaging system. Unfortunately, it is one of the most poorly documented areas of Windows, deserving much more attention than it gets. Writing a robust Windows program demands a good understanding of how your program and Windows communicate, and this is done via the messaging system -Windows sends messages to your program and your program responds, and vice-versa. This is all done via the message queue.

There are two types of message queues: a) system harware event queue; and b) application message queue. The former is for the hardware (e.g., keyboard, mouse) and timers. The latter is for the applications. There is only one system queue, but there is an application queue for each running program. So, when your program is running and it generates a message from Windows (e.g., the user changed the active window, generating a WM_PAINT message), Windows will post that message on the queue for your application, and on no other queue.

Every message is serviced by a function declared in one of the following two formats:

```
LONG FAR PASCAL Function(HWND hWnd, unsigned msg, WORD WPARAM, LONG LPARAM);

BOOL FAR PASCAL Function(HWND hWnd, unsigned msg, WORD WPARAM, LONG LPARAM);
```

The first is for registered dialog and/or window classes, and the second is for internal class dialogs. The parameters for both are the same: hWnd provides a handle to the window (telling Windows where the internal info for that window is in memory), msg is the message, and WPARAM and LPARAM contain additional information about the message (e.g., parameters). For example, a common message Windows sends is WM_COMMAND. This can be generated by the user clicking on a control (e.g., a button) in the window. For your program to know which control was selected, it needs the numeric identifier (every control in a window must have a unique identifier). The WPARAM parameter contains this information, and LPARAM is similar (as an aside, if you're wondering how to pass other data to and from the window's function (avoiding global variables), an excellent way is to use the "SetProp" and "GetProp" functions).

This is the work of the messaging system: it sends your program messages generated by Windows in response to the actions of the user running your program. Make sense? This leads to the "big switch" functions, so called because functions receiving these messages are typically set up with the first line as "switch (message) {", followed by a "case" statement (with code) for each message your program wants to act on. These functions can be several pages long.

After your program has received a message and acted on it, control should be returned to Windows. This can cause problems when you're trying to learn how to program in Windows. Most programmers are used to sequential code - the program is executed one line at a time. Under Windows, your program waits to get called by the messaging system. For example, let's say the user has just pushed a button in a window created by your application. This generates a WM_COMMAND message. Windows knows that the window belongs to your application, so the message is put in the message queue for your application (remember? each app has its own message queue). This "wakes up" your program, and control is passed to the function that handles the window that the user is in. When Windows gets to the line inside your "switch (message) {" block that reads "case WM_COMMAND:", it will usually find a line that reads "switch (WPARAM) {", which starts a new "big switch" block, which will have a "case" statement for each control in your window (it can handle other types of messages as well, such as menu items and accelerator keys). The "case" statements will get read until Windows finds the one with the identifier for the button that was pressed. Then, at long last, Windows will execute the code for the button pressed by the user. It's about time, isn't it? These things are never simple. Sure, I could rant and rave about how bizarre this whole thing is, but you could probably guess exactly what I would say, so it would all be kind of pointless. If you want to write a good, robust Windows application, you need to understand how all of this works, and it's not always intuitive what's going on. There's a lot of information a Windows programmer must keep in mind. If you don't understand (at a minimum) how Windows and your program are communicating, you're in trouble.

If you want a more detailed explanation of the messaging system, there are two good articles that may help clear it up:

- C User's Journal (May 1993) : "The Windows Messaging System", by Mike Klein; and

- Dr. Dobbs Journal (Feb. 1993) : "Inside the Windows Messaging System", by Matt Pietrik.

A Review of a Windows-hosted Programming Aid: Wilson WindowWare's WinEdit

By Jeff Perkel

Let me introduce myself. My name is "Jeff", and I'm a shareware junkie. For those of you who don't know, "shareware" is a marketing concept in which you are allowed to try a piece of software free, usually for 30 days, after which time you are supposed either to register the program, or delete it from your hard disk. It's really sort of an honor system thing. In most cases, even if you don't register, the only thing that happens is that you continue to get an annoying "Unregistered" message upon startup.

Because I am a shareware junkie, I'm always on the lookout for new and useful shareware. You'd be amazed at some of the truly wonderful utilities available out in netland for a fraction of what their
commercial counterparts cost. Why, I've got two text editors, a checkbook balancing program, a few games, and a few miscellaneous utilities - all shareware. So, after talking with Mike Wallace via e-mail, I've taken it upon myself to try and alert the readers of the WPJ to new shareware programming utilities. You may recall that last month (issue 5), a product called WinDisassembler was reviwed. This column will be similar to that. And now, without further ado . . .

-=-

One of the small pet peeves I have is writing a Windows program in DOS. This is because, until recently, I had to code in DOS, compile, and then run Windows to see that I have made an error, and then return to DOS. I know I could have done this in a Windows DOS session, but it's much easier, I thought, just to compile in DOS. Not to mention faster.

I currently am using the Microsoft C/C++ 7.0 compiler with the Windows 3.1 SDK. This package comes with an Integrated Development Environment (IDE) called the Programmer's Work Bench. This IDE runs from DOS, looks like DOS's EDIT.COM, and is, I thought, difficult to use. I wanted to be able to write my programs in Windows, in order to have access to my on-line SDK help, and, later, to the Developer Network CD.

Not having a lot of money to spend on this, I looked for a shareware utility that would do the job. I found one in Wilson WindowWare's WinEdit (reviewed in PC/Computing, April, 1993, page 129). This is a very useful programmer's aid. The key features are:

- Context-sensitive help. You double click on a function, such as CreateWindow(), and WinEdit will search through the Win3.1 SDK on-line help reference to find that topic. Not even Microsoft's Visual C++ Visual Workbench offers that feature.

- Multiple open files. Limited only by your available memory.

- Ability to launch a program from WinEdit. You can actually launch any program from WinEdit, but the Menu Bar contains a Project sub-menu, containing Compile, Make, Remake, Debug, and Execute functions. You can configure the command lines however you wish, and even save multiple configurations. For example, you might have a "windows" configuration, and a "dos" configuration, in order to compile using different command line switches.

Unfortunately, I was never able to successfully launch CodeView for Windows from within WinEdit - out of memory. However, I've also only got 4MB of RAM (for that matter, I couldn't launch CVW from the PWB, either!) I suspect, however, that on a better programming platform (i.e., 8MB RAM or more) that this would not be a problem.

- Compiler capture. The program captures compiler output, and moves you to the line with the first error. You can then skip to the next error, or to the previous error. In addition, you can view the compiler output itself.

- Macro language in the Professional edition.

- On-line help. Very comprehensive.

- Toolbar. This is not user-configurable, but just about any feature that you could ever need is already in there.

- User created extension DLLs. This allows you to modify WinEdit to suit your needs. The on-line help explains the implementation of these DLLs, as well as the WinEdit API.

I think that WinEdit offers us about all that one could ask for in a programmer's text editor. It's compiler capture feature and context sensitive help features make it a program that is certainly worth your while to examine - especially if you, like me, were laboring in the pit of DOS unnecessarilly.

-=

WinEdit comes in three versions: Lite, Standard, and Professional, all in the same ZIPped file.

The registration fees are $30, $60, and $90, respectively. However, only the standard and professional versions can compile. The professional version is simply the standard version plus a macro language.

Registration nets you a disk, a dialog editor, a printed manual, and shipping in the US and Canada.

WinEdit is available from PCCONTACT on Compuserve. Type GO PCCONTACT. WinEdit is in library 3 (utilities/misc) as WINEDT.ZIP. It is also available via anonymous ftp on SIMTEL20 and its mirror sites as pd1:< windows3>WE20N.ZIP.

[The author can be contacted at "perkel_j@a1.mscf.upenn.edu" on the Internet, and on CompuServe at 71404,123.]

Date: 02-Jun93 16:36 EDT
From: Mike Strock >INTERNET:MikeS@asymetrix.com
Subj: FW: White House Announcement About E-Mail (fwd)

From: Brian Granowitz
To: Cool (The Suave Group)
Subject: FW: White House Announcement About E-Mail (fwd)
Date: Wednesday, June 02, 1993 12:25PM

THE WHITE HOUSE

Office of Presidential Correspondence

For Immediate Release June 1, 1993

LETTER FROM THE PRESIDENT AND VICE PRESIDENT
IN ANNOUNCEMENT OF WHITE HOUSE ELECTRONIC MAIL ACCESS

Dear Friends:

Part of our commitment to change is to keep the White House in step with today's changing technology. As we move ahead into the twenty-first century, we must have a government that can show the way and lead by example. Today, we are pleased to announce that for the first time in history, the White House will be connected to you via electronic mail. Electronic mail will bring the Presidency and this Administration closer and make it more accessible to the people.

The White House will be connected to the Internet as well as several on-line commercial vendors, thus making us more accessible and more in touch with people across this country. We will not be alone in this venture. Congress is also getting involved, and an exciting announcement regarding electronic mail is expected to come from the House of Representatives tomorrow.

Various government agencies also will be taking part in the near future. Americans Communicating Electronically is a project developed by several government agencies to coordinate and improve access to the nation's educational and information assets and resources. This will be done through interactive communications such as electronic mail, and brought to people who do not have ready access to a computer.

However, we must be realistic about the limitations and expectations of the White House electronic mail system. This experiment is the first-ever e-mail project done on such a large scale. As we work to reinvent government and streamline our processes, the e-mail project can help to put us on the leading edge of progress.

Initially, your e-mail message will be read and receipt immediately acknowledged. A careful count will be taken on the number received as well as the subject of each message. However, the White House is not yet capable of sending back a tailored response via electronic mail. We are hoping this will happen by the end of the year.

A number of response-based programs which allow technology to help us read your message more effectively, and, eventually respond to you electronically in a timely fashion will be tried out as well. These programs will change periodically as we experiment with the best way to handle electronic mail from the public. Since this has never been tried before, it is important to allow for some flexibility in the system in these first stages. We welcome your suggestions.

This is an historic moment in the White House and we look forward to your participation and enthusiasm for this milestone event. We eagerly anticipate the day when electronic mail from the public is an integral and normal part of the White House communications system.

President Clinton	Vice President Gore
PRESIDENT@WHITEHOUSE.GOV	VICE.PRESIDENT@WHITEHOUSE.GOV

Text Manager Review

By Mike Wallace

Windows programmers get touchy when it comes to editors. All of the best editors, it seems, are DOS-based, and it's a pain to write the code under DOS, compile, and then start up Windows to test your app. A lot of time could be saved if there was a good Windows-based editor, but there don't seem to be very many to choose from. One choice is Text Manager. I've been a beta-tester for it, and it's a good program. Version 1.0 has been around for a while, but Digital Software is about ready to release version 2.0.

Text Manager can be used to edit any text file, but has added functionality for C files, INI files and batch files. If you select "Open..." from the "File" menu bar item, you can have it only list files with a filetype of ".C", ".INI", ".BAT", ".TXT" or ".*" to get all files. This comes in handy when you're writing C code. You can also open multiple files, and if you minimize the file, the icon for it is determined by the file type, so a ".C" file is readily distinguishable from a ".INI" file. Other features include:

- Toolbar and status bar

- Pressing on a menu bar item or button will cause the status bar to tell you what it does

- You can set the printer font to be different than the screen font

- Closing Text Manager when it is either maximized or minimized will cause it to open the same way the next time you run it (the author tells me this also is true for any text file you open, but I wasn't able to get it to work).

- Text Manager will remember the files you worked with the last time you were running Text Manager

- A full online help file (the author is working on this part now so I haven't seen it)

- Print preview

- Standard editing capabilities (e.g., cut, copy, paste, find, replace)

- Support from Digital Software - the author (Kerim Erden) checks his CompuServe mail several times a day and always answers all of his mail. All of my contact with Mr. Erden has been by e-mail and he has always responsed very quickly to my questions.

- Registration will net you an install disk (the installation program is graphical) with some extra utilities on it.

If you're looking for a straightforward, easy-to-use Windows-based text editor, try out Text Manager. It could be well worth your time to try it out. It's available as shareware and only costs $10. Text Manager 2.0 will soon be available on the IBMSYS, WINSHARE and NORUTL forums on CompuServe. Questions can be directed to Kerim Erden of Digital Software at CompuServe ID 72133,257.

Getting in touch with us:

Internet and Bitnet:

HJ647C at GWUVM.GWU.EDU -or- HJ647C at GWUVM.BITNET (Pete)

GEnie: P.DAVIS5 (Pete)

CompuServe: 71141,2071 (Mike)

WPJ BBS (703) 503-3021 (Mike and Pete)

You can also send paper mail to:

Windows Programmer's Journal
9436 Mirror Pond Drive
Fairfax, VA   22032
U.S.A.

In future issues we will be posting e-mail addresses of contributors and columnists who don't mind you knowing their addresses. We will also contact any writers from previous issues and see if they want their mail addresses made available for you to respond to them. For now, send your comments to us and we'll forward them.

The Last Page

by Mike Wallace

Got this in the mail a few days ago. Hope you like it. If you read this column last month (it was about OS/2) and want to read a different point of view from reader Jeff Miller, go to the "Letters" section at the beginning of the issue. Jeff's letter is included (in its entirety), followed by my response. This is the kind of debate I was hoping this column would spark. If anyone else has some criticism about this column or any other part of the magazine, please write. We want WPJ to be one of your sources of help when you're programming, and if we can make WPJ better, then we want to know how. Talk to you next month.

Date: 24-May93 11:31 EDT
From: Mike Strock >INTERNET:MikeS@asymetrix.com
Subj: NEWS BULLETIN - Men and women are NOT alike.

Sure, you thought you already knew that. But now we have proof! After countless hours of surveys and studies on the following topics, these facts have emerged.

Relationships:

Women have deep, meaningful, mutually nurturing relationships. Men have "that time when me and Suzie was doing it on a semi-regular basis".

When a relationship ends, a woman will cry and pour her heart out to her girfriends, and she will write a poem titled "All Men Are Idiots". Then she will get on with her life, usually by meeting another man and doing an emotional belly-flop.

A man will call six months after the break-up, at 3:00 a.m. on a Saturday night, and say, "I just wanted to let you know you ruined my life, and I'll never forgive you, and I hate you, and you're a total floozy. But I want you to know there's always a chance for us". This is known as the "I Hate You/ I Love You" drunken phone call, that 99% of all men have made at least once.

Women prefer 3040 minutes of foreplay. Men prefer 3040 seconds of foreplay. Men consider driving back to her place as part of the foreplay. Women consider getting married as part of the foreplay.

Maturity:

Women mature before men do. 17-year-old females can function as adults, but don't. 17-year-old males can't function as adults, and don't care. This is why high school romances rarely work.

Handwriting:

Women use scented, coloured stationery and they dot their "i's", with circles and hearts. Women use ridiculously large loops in their "p's" and "g's". It is a royal pain to read a note from a woman. Even when she's dumping you, she'll put a smiley face at the end of the note. Men do not decorate their penmanship. They chicken-scratch.

Bathrooms:

A man has six items in his bathroom: a toothbrush, toothpaste, shaving cream, razor, a bar of Dial soap, and a towel from the Holiday Inn. The average number of items in a typical woman's bathroom is 437.   A man would not be able to identify most of these items.   A woman uses each of them every morning.

Groceries:

A man waits till the only item left in his fridge are half a lime and a Blue. Then he goes to the grocery store, and buys everything that looks good.   By the time a man reaches the checkout counter, his cart is packed tighter than the Clampett's car on Beverly Hillbillies.   Of course, this will not stop him from going to the 10-items-or-less lane.   When a woman does this, it is called shopping.

Going out:

When a man says he is ready to go out, it means he is ready to go out. When a woman says she is ready to go out, it means she WILL be ready to go out, as soon as she finds her earring, finishes putting on her makeup...

Cats:

Women love cats.   Men say they love cats, but when women are't looking, men kick cats.

Dressing up:

A woman will dress up to go shopping, water the plants, empty the garbage, answer the phone, read a book, get the mail.   A man will dress up for: weddings, funerals.

David Letterman:

Men think David Letterman is the funniest man on the face of the earth. Women think he is a mean, semi-dorky guy who always has a bad haircut.

Laundry:

Women do laundry every couple of days.   A man will wear every article of clothing he owns, including his surgical pants that were hip about eight years ago, before he will do his laundry.   When he is finally out of clothes, he will wear a dirty sweatshirt inside out, rent a U-Haul and take his mountain of clothes to the laundromat.   Men always expect to meet beautiful women at the laundromat.   This is a myth.

Weddings:

When reminiscing about weddings, women talk about "the ceremony". Men talk about "the bachelor party".

Socks:

Men wear sensible socks.   They wear standard white sweatsocks.   Women wear strange socks.   They are cut way below the ankles, have pictures of clouds on them, and have a big fuzzy ball on the back.

Slippers:

Men wear leather slippers that have been brought to them by their faithful dog.   Women wear huge fuzzy orange things with faces, that look like puppies.

Nicknames:

If Gloria, Suzanne, Deborah and Michelle get together for lunch, they will call each other Gloria, Suzanne, Deborah and Michelle.   But if Mike, Dave, Rob and Jack go out for a brewsky, they will affectionately refer to each other as Bullet-Head, Godzilla, Peanut Brain and Useless.