The Stanford/Palo Alto Users Group for the IBM PC

PO Box 3738 Stanford, CA 94305

NEWSLETTER VOLUME 1 NUMBER 5 AUGUST 1983

<<<< THERE HAS BEEN A ROOM CHANGE FOR THIS MEETING >>>>
(what a surprise)

Please note that we are meeting at the Graduate School of Business on Stanford Campus.

```
*****************************************************************
*  Next meeting:        Wednesday, August 31, 1983          *
*                       7:00pm                              *
*                       Rm 54, Graduate School of           *
*                       Business, Stanford University       *
*****************************************************************
```

AGENDA FOR THE MEETING —

7:00 General Club Business
    Report on the Monday before the Wednesday Meeting
    New Business

7:40 Guest Speaker John Horan. John is writing a book on dBase II, and will speak and field questions concerning database management.

8:45 Random Access

9:00 Conclusion of the Official Club Meeting

Reminder—Don't forget about the Monday Before the Wednesday Meeting at Talbott's on California Ave, across the street from the Winery. Time is at 8:30 pm, goes on until whenever (usually about an hour and a half or so). All are invited to this meeting where great plans are made and policy discussed. We will be discussing what guest kers to invite and what specific topics we would want focus on in future meetings.

It goes without saying that move pretty fast in the silicon lane. Sometimes I think that things change just for the sake of.....well, you know how it goes. In that spirit we have a new official post office box at Stanford, Box 3738. Why? If bigger is better, then we're on our way, because our new box is twice the size as the old one. Large enough, in fact, to put a mini floppy diskette into without folding, as was the case with our old box. I tried to get them to set the combination to I-B-M but they wouldn't go for it.

FROM THE LIBRARY

Stuffed into our old mailbox the other day were two diskettes, both of which asked us to add their contents to our club library. No, it wasn't a personally autographed floppy from Peter Norton, but one of them is a "well presented Educational game, that teaches history whilst having fun. These programs were the best written by my pupils for their Basic finals. They have agreed to share them with the rest of the IBM-USER groups..." You will need to copy DOS and BASICA onto the diskette. The other diskette has a label with "A>PIZZA" on the front. It contains all the information you would ever need to set up pizzeria, complete with five kinds of sauces, and coincidentally wants to sell you a $200 list management program.

PERIODICAL REVIEW

A slow month for review, as Byte and PC Age and Softalk weren't on the newstand yet. Who says there's no rest for the weary.

Dr. Dobbs' Journal has an article on building a serial-to-parallel convertor, complete with schematics. Also a 56 word text editor program.

PC World has a very good article on hard disk drives, worth reviewing or at least hanging on to if a meg or two is in your future. PC-File, the freeware database program (which we have in our club library) is reviewed in the September issue of PC World. Andrew Fluegelman has a checkbook ledger routine for Lotus 1-2-3. There is a review of Spinnaker Software's recent educational programs for ages 3-10.

In the September issue of PC, there is a review of: The Pascal MT+86 System from Digital Research; a couple of speech synthesizers; and TK!Solver, from the inventors of Visicalc. There is an excerpt from a reasonably good book on writing programs that take advantage of the 8087 co-processor chip. Peter Norton starts his new column. Somehow, PC Magazine is looking more and more like Time Magazine. Oh, I almost forgot. A quick overview of ten programming languages (in English) starts on page 110. If you read all about them you may look good at the next cocktail party, or at least convince your friends that spending $5000 has in fact made you smarter.

NOTES FROM THE LAST MEETING....

The general membership voted to change the format (and price) of club membership. Be it known that one full year's membership now costs $25, which entitles the mem- to this illustrious newsletter, and unlimited access t the club library. Those who have paid past fees ($10 or $15 family membership) will receive credit for that amount toward the $25 dues. The $25 fee covers each household; that is, there is no longer a separate fee category for families. This will be the last newsletter sent to those people who don't make up the difference by this meeting (8-29-83).

UNERASE (of sorts)-Do it yourself! by Paul Simon

This article was originally written so that I could share what I thought was a clever way to undo the damage done after accidentally erasing a file. The way to avoid that sinking feeling was originally written in PC World, and it seemed like a good idea at the time (the downfall of all great minds). If you have the current issue of PC World (August, p566) you will notice an addition which basically tells you it ain't so simple. However, if your file is less than one sector long, that is 512 bytes, it will still work. Let us continue.

For those of you who have not yet explored that part of DOS except for the DOS operating commands themselves, here is an interesting opportunity to find out more about your machine and at the same time perhaps do yourself some good when you accidentally erase a file and reg it. The commands in DEBUG certainly appear mysterious but with a little work on your part there is a payoff, as we will see how to "unerase" a file. There are no commands such as "unerase" in the collection as you might have already noticed, either in DOS itself or in DEBUG. There is a way however, to read a file in a very explicit way, put it into memory, examine the contents of memory (actually what was on the disk), change it if you wish and then write it again to disk. It actually can be fun!

Look at the commands in DEBUG called "load","dump", "edit", and "write". In the usual and rather dry IBM style they describe how to load information from the disk to memory, how to read memory, how to edit that information and how to write it back to the disk. Of course this can be dangerous to your data so do your experimenting with a copy of your valuable disk.

For some background on what we will be doing, first read appendix C in the DOS 1.1 book. This describes the way that the file directory is made and stored on the disk. There is a difference in single sided vs. double sided disks but that is relatively minor. The directory can be found at track 0 sectors 4-7 (and in addition for double sided disks, sector 8, and the other side, side one, track 0, sectors 1 and 2). What we will do is copy the contents of the file directory into memory, examine it at will, change it to "unerase" a file and then write corrected directory back to the disk.

When I first looked at these commands I was concerned that I did not understand the memory addressing in the PC, but quickly found out that DEBUG keeps pretty good track of what is going on. Addresses are described as a "segment" plus an "offset", with the last hex digit dropped off the segment address. Thus the address 4b50H is written as 4b5: when it represents a segment address. The offset is merely added to the segment address so that 4b5:100 actually is 4b50 plus 0100 hex or 4c50. You will find it easy to use these ideas since the registers keep track of segment addresses and all you have to handle are offsets. The examples will clear this up.

The function of the "load" command is to transfer data from disk to memory, "dump" to examine the contents of memory in hex and ASCII, "edit" to change the contents of memory, and "write", to write from memory to disk. If you look at the "load" command you will see that addresses are optional. If no address is used , DEBUG will start to load into memory address CS:100 for you. That address is the segment address stored in the Command Segment register, to which is added the offset of 0100. This is 100 bytes above the start of useful memory. We will also have to say from where on the disk we want to read that data, so we will also need the starting sector and number of sectors we want to put into memory starting that address, CS:100. The absolute address is the ment address stored in the CS(Command Segment register) plus an offset of 0100H. In all command formats, the memory address is optional. DEBUG will keep track of the address for you, and it uses that address for all commands.

DEBUG is on the DOS disk, so place it in one drive and the disk you wish to look at in the second drive. There are other ways of doing this of course. You can DEBUG the DOS disk (not recommended!!), or start DEBUG and place the second disk in drive a. I find it easier to boot up DEBUG in drive a and load from drive b.

Type "DEBUG" and return and you will get a "-" prompt followed by the flashing cursor. Then type the following";L 0 1 3 3. This will read from drive "1" (otherwise known as drive b), starting from sector 3, three sectors, and place the contents into memory starting at CS:0000. There are some strange inconsistencies in the naming and numbering system here. First, as you have noticed, the drives now have numbers instead of letters. Secondly, sector numbers now start at zero, whereas the description on page C-1 numbers from 1 (one). Also note on page 6-8 that sectors are numbered as far as DEBUG is concerned from 0 to 13F, and if it is a two sided disk, continuing 140 to 27F.

DEBUG has kept track of that CS segment address and now we can look at what we have done by using the dump command and typing "d" at the DEBUG prompt. The same result obtains with "d:0", or with whatever offset we have used. What you now see is the disk directory as written, and it includes the two hidden files, IBMIO.COM and IBMDOS.COM. To continue looking in the directory, continue to type "d". If you lose your place you can add the offset to "d". There are 512 bytes per sector, and so four blocks of 8 by 16 bytes of dump per sector. We have copied three sectors or twelve blocks of dump to examine. You will find some other surprises when you scan down the directory. Lo and behold there are those files you erased but the names look funny. The first character has been changed to e5H. And if you continue looking down the directory the empty entries also start with e5H. That is what signals DOS that the entry is to be ignored. When you "erase" a file DOS replaces the first character of the filename with the hex byte e5. Unfortunately it also makes some rather nasty changes in the file allocation table (FAT). If you have not written anything else since you erased that file it is possible to unerase it then by replacing that e5H with a valid character, such as an ASCII "B", 66H. Write using the DEBUG write command to place the corrected data back in the file directory area on the disk. I'm chicken so I use the absolute address in the format for the write command, in my case 4b5:000 1 3 3.

I hope this small exercise will encourage you to explore the innards of the PC especially with some dividends if you should have need of "unerase". You will have real sense of what contiguous files are in contrast to fragmented ones and the value of compressing a seemingly full disc by use of the *.* copy command. The change is dramatic in the file directory. Have fun!

Questions or comments?

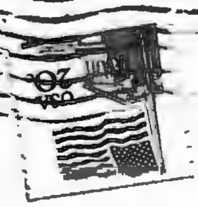
Kevin Ohlson	Newsletter editor	494-2574
Linda deSosa	Membership	856-6281
Wes Danskin	Mystery Treasurer	851-0277
Mike Van Waas	Librarian	325-2507

Monterey

$675 reduction.

246-3500

SAN FRANCISCO, CA
PM
AUG 26
1983
941



The Stanford/Palo Alto Users Group
for the IBM PC
PO Box 8292
Stanford CA 94305