OPCUG The Ottawa PC Users' Group

Mailing Address - 3 Thatcher Street, Nepean, Ontario K2G 1S6
(613) 723-1329 (voice) (613) 565-1880 (BBS - N,8,1)

April 1991
Vol. 8 No. 4

MEMBERSHIP RENEWAL TIME

For most of us it is membership renewal time. If the expiry date on your mailing label is 91-03-31, please complete the form on the back of the Newsletter and bring it to the meeting. If you are unable to attend, send the form and a cheque to The OPCUG, 3 Thatcher Street, Nepean, Ont., K2G 1S6.

For those new members who joined in October 1990 or after, your membership runs for a period of 12 months. If you believe that the expiry date on your label is incorrect, please contact Harald Freise at 828-3411.

SPEED

By John Whelan

I've been hearing quite a bit of talk about 286, 386SX, 386 and 486 machines, some of which makes sense and some doesn't, so I thought I'd scribble a few lines down about the subject.

If we go back in time to the days of the CP/M, the Z80 and 8080 chips dominated the market. Working Micros were sold with 2k of memory, and some systems even had 64k of memory although no one could find a use for it all.

The IBM PC was based on a different Intel chip the 8088 which offered direct access to 64k of memory but had an indirect way of addressing different banks of 64k. In fact the 8088 chip could address 1024k of memory and, in the IBM PC, it was allowed to use 640k for programs and data. The 64k limit still lives with us today, programs such as Turbo Pascal's built in editor will only edit a maximum file size of 64k. I think GW BASIC again has a 64k limit. The programs that need to exceed this limit use additional commands and more complex logic. Expanded memory can be accessed beyond the 1024k but that really uses a convoluted way of switching the contents of memory around so that the CPU can get to where it wants to be.

The 80286 has two modes: protected and real. I think they are called. It can in one mode by use of additional instructions access memory beyond the 1024k limit but again it uses indirect instructions that add complexity to the program.

The 386SX, 386, and 486 are all basically the same processor that can run the same instruction set. This instruction set is a super set of the 80286 set, but it has some additional capabilities. These are better memory management, little things like most of these machines when running programs such as Windows 3.0 can use virtual memory, ie if the program needs 8 megabytes and you only have 2 no big deal it will just use 6 megabytes of hard disk instead. Also you'll often find software utilities that can remap the memory above 1024k from extended memory to expanded memory and back again, this can be very useful. Your program runs a little slower but the operating system takes care of it all.

The other major difference is in the way that programs can access memory. Instead of only being able to address a 64k segment directly, the program can address many megabytes, the exact number escapes me but it is more than sufficient to be able directly address any RAM on a PC. These differences mean that it is much easier to port mini and mainframe programs to a 386, and more complex programs can be more simply written. Already we are seeing programs that will only run on a 386. The SX will run all the programs a 386 will but often a little slower, the 486 is simply a faster version of the 386.

Ah, but a 95 Mhz 286 is faster than a 386SX I hear from the back row. Well, yes and no. If I have a program that only needs 286 instructions, then a 386SX or 386 will often run no faster. However, if you run programs such as Pkzip or Pkunzip, you'll notice the 386SX will run faster. When trying to run a program that only runs on a 386, speed just isn't relevant.

The 386SX, 386 and 486 all offer greater flexibility today than the 286 CPUs. However only Intel makes them, the 286 CPUs are made by Intel, Harris and other companies so just sometimes you'll see advertising aimed to keep you buying 286 CPUs.

Finally I'd just like to remind you that the humble 8088

(*cont'd on Page 2*)

CONTENTS

PC is still a very capable machine. It still runs Telix, WordPerfect, Turbo Pascal, BASIC and a host of other programs.

The latest thinking of File servers for LANS seems to be that the CPU speed is not as important as was once thought and that the hard-disk access time and bus speeds play a much larger role in giving good performance.

*** * ***

IS THIS THE RIGHT DISKETTE?

By Robert Parkinson

The tip for this came from a recent computer magazine, but for the life of me, I can't find it again to cite it here. Oh well!

How many times have you, when in a hurry, put in the wrong diskette and had your program or batch file crash in a dismal heap with a "file not found" or "bad command or filename" message? How about setting up a batch file so that youngsters can stick in the diskette to run their favourite games? There are utility programs around that will check the diskette volume label and return either the label name or an errorlevel based upon a designated target label. However, these tend to be a bit unwieldy to use in a simple batch file.

There are several techniques using the DOS CHKDSK command to do this as well. These rely on using CHKDSK to "pipe" the volume label through the FIND filter and then redirecting the result to a new temporary batch file (e.g. TMP.BAT). TMP.BAT, in turn, simply invokes the permanent batch file with the same name as the volume label. If you are interested in this convoluted approach, look at PC Magazine Vol 8 #1 (17 Jan 89) on page 346.

But, here's another way to do the checking, in a very simple manner.

First, create an identifying zero-byte file on the diskette you want to identify. The reason for a "zero-byte" file is that it only takes up a directory entry on the diskette, not a data cluster. Let's say it is the SMALLGAMES diskette. To do this, enter the following from the DOS command line, changing the drive and/or identifying filename to suit yourself ("!utility.054", for example):

"If not exist b:\!smallga.mes echo >> b:\!smallga.mes"

If that's too much to type, enter this from the command line, ignoring the inevitable "file not found" message (it works anyway!).

"Type xyx.fil > !smallga.mes.mes"

Now you have your diskette identifier, the zero-byte file "!smallga.mes". I used the exclamation mark as the first character so that this filename will always appear first in any directory listing.

To use it, change your batch file to resemble something like the following example:

```
@echo off
cls
echo      Insert the SMALL GAMES diskette in Drive B: and
pause
if not exist b:\!smallga.mes goto ERROR
echo      This is the SMALL GAMES diskette.
....
(remainder of commands go here)
....
goto END
:ERROR
echo      This is NOT the correct SMALL GAMES diskette.
echo      Change the diskette and start again.
:END
```

A "quick and dirty", but quite effective, solution to a vexing problem. Compliments of the "unknown magazine".

*** * ***

OOPS!
WP5.0 LOSES SCREEN IMAGES

By Doug Poulter

I'm a heavy WordPerfect user and am quite gung-ho about the product. This means that I regularly find little malfunctions as I stretch its capabilities to their fullest. Here are two of my latest discoveries. As far as I can tell, they only apply to the two most recent interim releases of WordPerfect 5.0, dated 05/30/90 (US) and 06/30/90 (Canada).

Both malfunctions have occurred only in documents with multiple comments, all within a few codes of each other. Both cause the screen formatter to misbehave while attempting to display multiple comments. Neither malfunction corrupts the document produced, and neither exists in WordPerfect 5.1.

The first error occurs when the arrow keys are pressed. The screen formatter occasionally blanks out the screen completely except for the blinking cursor. When this occurs, you can recover the screen image by pressing <Ctrl F3> twice (which forces a screen "rewrite") and waiting a few seconds. Occasionally I've found it necessary to repeat this procedure to correct the problem.

The other malfunction occurs when the cursor is repeatedly moved back and forth over multiple comments, all within a few codes of each other. Text lines within the comments shift left until they are partially off the screen and, in so doing, overwrite the left side of the comment box. Display of the comments may then become corrupted. To fix this problem, press <Page Up> or <Page Down> several pages, and then return to the comments. They will be formatted correctly. My guess is that this causes the screen formatter to erase its buffers and redraw the screen and the comments.

While WordPerfect (like its competition) is not bug free, the Corporation is the most user-friendly company I've had the pleasure of dealing with. I use their toll-free numbers to call in to report bugs, and they regularly send me free updates as soon as interim releases correct them. Many users also call in for free technical support when they encounter problems that leave them scratching their heads. The staff on the other end of the line are knowledgeable and friendly.

While Mark Baker ended his article of the February Newsletter with a comment

that Microsoft Word is superior to WordPerfect in many ways, the reverse is also true. User support and product functionality are two cases of the latter.

* * *

PROLOG
Predicting with Examples

By José Campion

(Cont'd from March 1991)

Now let's modify that program so that it can output all possible road connections while indicating middle towns in the route:

```
% Example 2.-
% -----------
domains
  town     = symbol
  distance = integer

database
  road(town, town, distance)

predicates
  assert_db
  nondeterm route(town, town,
    distance)
  nondeterm route(town, town, town,
    distance)
  get_route_direct
  get_route_indirect

clauses
  assert_db:-
    assert(road(tampa, houston, 200)),
    assert(road(gordon, tampa, 300)),
    assert(road(houston, gordon, 100)),
    assert(road(houston,
      kansas_city, 120)),
    assert(road(gordon,
      kansas_city, 130)).
  route(Town1,Town2, Distance) :-
    road(Town1,Town2, Distance).
  route(Town1,TownX,Town2, Distance):-
    road(Town1,TownX, Dist1),
    road(TownX,Town2, Dist2),
    Distance=Dist1+Dist2.
  get_route_direct:-
    route(From, To, X),
    write(From," - ",To," (direct).
      Dist = ",X),nl,
    fail.
  get_route_direct.
  get_route_indirect:-
    route(From, Via, To, X),
    write(From," - ",To," (via ",
      Via,"). Dist = ",X),nl,
    fail.
  get_route_indirect.

goal
  assert_db,
  makewindow(1,31,31,
    "Route window",5,10,15,60),
  get_route_direct,
  get_route_indirect,
  readchar(_),
  removewindow.
```

There should be few mysteries for the attentive reader. There are now two road predicates declared with different arities. The word "nondeterm" precedes the declaration of both. This is a way to indicate to the compiler of PDC Prolog that we know that the clause produces more than one result (it is "indeterminate") and that we like it that way and that it should stop insisting with its erudite warnings. Actually, these warnings are extremely helpful when indeterminacy results from an error or oversight from the programmer. In these cases, the cut is usually used to solve the problem (the cut, always the cut...!, and you haven't seen anything yet..., at the end of the following paragraph cuts are coming in glorious Technicolor...)

Example 2 also includes the "fail" predicate. Guess what... this predicate always fails. Why do you want to always fail? neither for the sake of it nor for masochism, but to trigger backtracking, of course. In example 2 fail is used to cause the program to explore alternative routes. Why then the second (empty) call to each rule?:

```
    get_route_direct:-
      route(From, To, X),
      write(From," - ",To,
        " (direct). Dist = ",X),nl,
      fail.
->  get_route_direct.              <-

    get_route_indirect:-
      route(From, Via, To, X),
      write(From," - ",To," (via ",
        Via,"). Dist = ",X),nl,
      fail.
->  get_route_indirect.            <-
```

The reason is to prevent failure in the calling clause, in this case the goal itself. If the first call to the rule fails then Prolog backtracks to the second call of the same rule which succeeds instantaneously and the "failure" is not carried to the goal.

Example 2 also uses "nl" to introduce the carrier return character and uses the predicate makewindow and remove window. This, of course, is specific of PDC Prolog very powerful screen capabilities. Makewindow has an parity of 8, the different arguments indicate the window number, the attribute of the field within the window, the attribute of the border, the title of the window to be inserted at the middle of the top frame, the starting row, the starting column, the number of rows and the number of columns. PDC Prolog version 3.2 introduces another makewindow predicate of arity 11 which allows a the user to define the border, the position of the title and whether the screen should be refreshed after making the window. Predicate removewindow is used to remove the window made last and the cursor is re-positioned in the window that was active preceding that call. PDC Prolog claims that a maximum of 34 windows can be successively overlaid in a given screen. Experience has shown that (unless my copy of PDC Prolog is faulty) this is not true: the maximum is 42!

Last (but not in the least) there is this call to readchar(_). Prolog implements reading predicates specific of the different kinds of domains to which the character being read may belong: readchar reads a character, readint reads an integer, readreal read a real, there is even a special predicate "file_str" that can read a whole text file into a single string of up to 64K characters! (Doesn't Turbo Basic do that too...?) There are also many predicates for conversion between domains. Both examples have included the simplest ways of writing on the screen, but there is a "writef" predicate for formatted output. But coming back to readchar(_), what is readchar trying to read? worst, where is it trying to read from or to read into? Well, if the input device is not specified PDC Prolog assumes that you are referring to the keyboard (the same is true for the screen as the output device). So readchar will try to read a character typed from the keyboard into the variable between brackets, only... that this is not a valid "variable" (it does not start with a capitalized letter, remember?). What is it then? Prolog uses the underline (_) character to indicate an "anonymous"

variable which is the Prolog way to say "anything goes". Thus readchar(_) is equivalent to Pascal's "ch:= readkey;". The PDC Prolog compiler does not like the existence in a clause of variables which are not used; yet in many instances a rule will only need to use a subset of the arguments in the arity of a term. In this case all unused arguments must be represented by the "anonymous variable" underscore (_) character.

Example:

```
  % Example 3.-
  % -----------

  domains
    lengua = symbol
    company = symbol
    version = real

    year = integer


  predicates

    nondeterm lng(lengua,
company,
            version, year)


  clauses

    lng(prolog, borland, 1.0,
1986).
    lng(prolog, borland, 2.0,
1989).
    lng(prolog,

      prolog_Development_Center,
        3.2, 1990).
    lng(pascal, borland, 3.0, 1983).
    lng(pascal, borland, 4.0, 1986).
    lng(pascal, borland, 5.5, 1989).

  goal
->   lng(Lang, Company,_,_),          <-
    not(Company = borland),
    write(Lang," (",Company,")"),nl,
    fail.
```

This example shows the use of the anonymous variable character. In addition notice the use of the "static" database represented by the repeated lang clauses. These terms are not asserted to any database in memory, they are accessed directly as part of the program code. This represents a "static database" because the terms cannot be upgraded or deleted at running time. The goal also implements the "not" predicate which succeeds when the condition within brackets fails (it causes instantaneous success out of failure). In the example it is requesting "any program except those from Borland". This may be difficult to accept by Borland, so, if your name is Philip Kahn, please delete "not" and all and only Borland languages will be output.

The word "language" was also avoided by the use of the lng functor. This was due to the fact that "language" is used to declare the compiler used to produce external predicates. As indicated in the introduction, PDC Prolog accepts external predicates in the form of OBJ files output by "C", Assembler or Pascal compilers. More about this later.

... to be continued.

THE DOS ENVIRONMENT

By Robert Parkinson

(continued from March 1991)

For example, my batch file to call the Telix communications program has a line at the beginning with the syntax LOGIMENU TLX.MNU and a line near the end reading LOGIMENU OUT. Some readers will have a pet TSR that is not amenable to this latter approach and will, perforce, have to adopt the former method.

A point of interest in regard to your AUTOEXEC.BAT file! Many of us like to add a > NUL at the end of lines to avoid the plethora of on-screen messages, such as copywrite notices, etc. This trick directs the screen output to the NUL device driver, DOS's garbage can. This is fine for all but TSRs. What happens when you do this with most TSRs is that the open file, the NUL, is not closed properly when the TSR exits to DOS, leaving you with a useless file locked into memory. Try it yourself by examining your memory with one of the utilities that displays open files, such as PMAP. So what? Well, the DOS (3.30) default for FILES is eight, meaning that if you permanently lock up five or six of them using NUL, you will not have enough left for some of your applications and will get a "Too Many Open Files" error message. Then you are forced to increase the number of FILES in your CONFIG.SYS file and each additional one costs you 48 bytes of RAM. You might think that the DOS internal CTTY command might solve the problem, using the syntax CTTY NUL, then the commands to load your TSRs, then CTTY CON. Not only is this is a bit perilous, as neither the screen nor the keyboard are operable between the two CTTY commands, but it won't work here. You end up losing two FILES for each TSR, one for NUL and one for CON.

Before I leave the matter of TSRs, let me give you one more small tip that may save you a few more bytes of RAM if you are really desperate for space. As an experiment, boot up your system from your hard disk with no AUTOEXEC.BAT file. If you then look at your environment with SET, you will find that you have two default entries in your Master Environment Block – PATH= (no following string) and COMSPEC=C:\COMMAND.COM. Since you don't need to set either of these variables until very near the end of your reorganized AUTOEXEC.BAT file, start out your new AUTOEXEC.BAT file with two new commands, SET PATH= and SET COMSPEC=. The environment copies given to your TSRs will now not include those useless strings. This simple action could save you several hundred bytes of RAM if you are loading 12 or more TSRs.

PARENT VERSUS CHILD ENVIRONMENTS

We have already briefly mentioned parent versus child programs and the fact

that every program must have a parent. Normally, the parent is COMMAND.COM, because most programs are executed by COMMAND.COM. We have also discussed the fact that the child application program, unlike the secondary command processor or shell, gets only a somewhat limited program copy of the environment to use and, because of the order in which DOS loads the components of the program, there is no way to expand that copy of the environment. This program copy given a true child application program, whether memory-resident or not, is only large enough to accommodate the current variables (again, rounded up to the next 16-byte boundary). While this is interesting, the average user will likely be more concerned about a child environment when shelling to a secondary command processor, and maybe not even then.

But there is one very important reason why you need to understand the parent and child relationship. Because any child program is only working with an environment copy, this copy disappears when the program terminates. Therefore, a child program, including a secondary shell processor, cannot normally alter the parent environment. This means that when you are working in a secondary command processor, such as shelling to DOS from an application program, you have no way of altering the DOS Master Environment Block. Is this really true? Well, DOS was designed that way as a safety measure, but there are ways around the restriction. Programmers often circumvent this barrier, but what about the average user?

Here I'll mention SUPERSET.EXE again. This excellent utility program, developed by Richard Linley of Kingston, has many merits. It has a number of built-in SET-like functions, such as date, day-of-week, time, current drive, current directory, and so on. These can be put into the environment with a simple command, either from the DOS command line or from within a batch file. It will also permit you to load an entire group of variables, say your entire environment, from an ASCII text file with just one simple command. The length of the name SUPERSET itself is too long for frequent use, so I simply renamed it to SSET.EXE. However, the real point that I want to make here is that SUPERSET (version 1.4 and later) has the unique capability of altering the parent environment. SUPERSET will accept multiple command line parameters, each of which can affect the environment. Using an initial @BOT parameter, all subsequent SUPERSET parameters on the same command line will take effect on the Master Environment Block until this feature is turned off with an @TOP parameter. Note that any parameters between the @BOT and the @TOP will not affect the active copy. For example, the command SUPERSET @BOT NAME=BILL @TOP NAME=JIM will put the variable NAME=BILL into only the Master Environment Block and the variable NAME=JIM into only the active environment. SUPERSET will warn you if you are working with a temporary copy of the environment so that you can use the @BOT capability if you wish.

I must also mention once again the desirability, if you are using a batch file, of using the CALL command (DOS 3.3 and above), instead of a secondary copy of COMMAND.COM. CALL does not involve a child process and therefore the real Master Environment Block may be altered from within a subordinate CALL routine.

I mentioned the subject of the Program Segment Prefix (PSP) a little earlier. While I said that we were not going to consider it further, nevertheless one anomaly deserves mention. The PSP contains the memory address of the parent. In the normal course of events, when one program calls another, the PSP of the child will show the address of the calling program as the parent. This allows tracing of the entire sequence. However, DOS doesn't always follow it's own rules. When you shell to DOS from an application program, the bootup COMMAND.COM (in some versions of DOS) puts it's own address in the PSP of this secondary copy, rather than the normal address of the calling program. Microsoft confusing us yet again! Anyone interested in further reading may consult Reference G.

Before leaving the subject of child versus parent programs, I should again briefly touch on the matter of nesting of actual application programs. I mentioned this earlier in the part on Expanding the DOS Environment under the Shell Command section. For most users this is of no concern at all. Even in the case where you have shelled to DOS from an application program so as to temporarily call up a second application, assuming that your available memory permits this, you are unlikely to be consciously using the environment at this time. Therefore I have put the remarks on nesting of application programs in Note #8 at the end of the article for you to read or not as your curiosity dictates.

ALTERNATIVE COMMAND PROCESSORS AND THE ENVIRONMENT

The phrase "Alternative Command Processor" has a most authoritative ring to it, but remember that it simply means a different command-line interpreter or an alternative user interface. I am not in a position to discuss these at any length. With one exception, I have not extensively used Windows, 4DOS, DoubleDOS, DESQview, or the other options to the DOS COMMAND.COM. Users should approach alternative command processors with some degree of caution until they learn how they interact with the DOS Master Environment Block, if at all, and whether the user's preferred environmental variables (say for WordPerfect) will function properly. For example, if you shell to a secondary command processor from an application that uses the COMSPEC variable to find the command processor, you are unlikely to notice any adverse changes as long as your COMSPEC now points to the location of the new alternative processor. However, some application programs are hard-coded to invoke COMMAND.COM alone as the secondary processor. Here you may run into serious difficulties.

The shareware command processor 4DOS is a most interesting program. Although I have not tried the latest version 3.01, previous versions already had greatly increased capabilities over the DOS COMMAND.COM. In addition to a large number of new internal commands and a much-enhanced batch file handling capability, 4DOS is quite flexible in the way in which it treats the environment. Normally, 4DOS swaps it's environment block to high memory or disk when not being used, which can crash unwary programs. However, by use of a start-up switch, you can direct 4DOS to create a standard environment block instead. In my limited experience with it, I have found no

problems with the way it handles the environment.

... To Be Continued.

PUBTALK

From: DANA WEBBER
To: MICHAEL GODDARD (Rcvd)
Subj: RBBS

What's the difference between a point and a bbs?? i thought that all points were bbs because they had to answer the phone.

From: MICHAEL GODDARD
To: DANA WEBBER
Subj: REPLY TO MSG# 17991 (RBBS)

A BBS frequently is running constantly and takes users or "mail" at any time
(in Ottawa, there is a "mail period" from 4-6 am when users are usually not allowed on). It does mean a dedicated telephone line.

A "point" is really a "super user" for a BBS. Points don't run constantly, they usually "poll" their "boss node" (the BBS they are associated with) daily to pick up mail. By using mailers, packets of mail flow quickly as there isn't all the signing-on and keyboard interaction at the start of a session. A point is obligated, however, to call in frequently. The Boss usually keeps a lot of mail for the point and if the point doesn't call in, it consumes more and more of the bosses' disk.

Points usually only contact the FidoNet world through their boss, again, as a kind of "super user". A BBS Sysop, however, can swap mail at will throughout FidoNet directly (in theory, but not really in practice).

From: DOUG HEWKO
To: JOHN WHELAN
Subj: REPLY TO MSG# 10227 (HELLO)

What is the difference between the abilities of Lotus 1-2-3 version 2.2 and Quattro Pro?

From: JOHN WHELAN
To: DOUG HEWKO (Rcvd)
Subj: REPLY TO MSG# 10240 (HELLO)

Purely personal but Quattro is much easier to work with, and doesn't have the same size restrictions as Lotus. It has much better printer support which means you can generally avoid the Lotus step of having to print to disk and then bring the file into WordPerfect to make the results fit the page or look pretty. It doesn't seem to get in the way of the numbers. Pick up doc17 and look up some of the reviews in the library and you'll get a much better idea.

From: DOUG HEWKO
To: DANA WEBBER
Subj: MODEM PHONE LINES

Does Bell Canada charge more than the standard rate for a line that will be used by a modem? I have heard that if they know you have a modem, they will start to automatically charge you extra. I do not know if there is any truth to this.

From: JOHN WHELAN

You can request a data line and that is what Bell Canada recommend for modems. Commercially the cost for a single phone line is $35 a month a data line costs about $60+ a month. In Ottawa there is basically no difference in line quality. Bell will however condition up a poor quality data line to a defined level of whatever. If you have problems with an ordinary voice line a complaint that you can't make out what your Grandmother is saying usually gets the same results.Most modern modems will tolerate a surprising amount of poor quality phone line. The V.42 on line 4 certainly copes with transatlantic phone lines without problems and as a rule of them they are some of the worst. I've connected to Yellowknife and other remote locations over the Government private network using MNP 5 without problems so voice grade should do fine. Bell does not start charging you a higher rate for having a modem on the line.

There has been some moves in the states to charge long distance modem users more than voice, this is because the modems are squawking all the time but voice has quiet patches when you can multiplex another conversation in on the same wire.

THE LAN TUTORIALS

By Aaron Brenner
(cont'd from March 1991)

Like LLC, it conforms to the OSI model. IBM's SDLC (Synchronous Data Link Control) is a Data Link layer standard that does not conform to the OSI Model but does perform similar functions. IBM has many products that do not follow the OSI Model or its hierarchical setup. IBM has pledged support of OSI, however.

Transport Protocols

The ISO is in the process of establishing protocol standards for the middle layers of the OSI Model. As of yet, none of these have been implemented on a widespread basis, nor has the complete OSI protocol stack been established. To make matters more confusing, most of the middle-layer protocols on the market today do not conform neatly to the OSI Model's network, transport and session layers. They were created before the ISO started work on the model.

The good news is many existing protocols are being incorporated into the OSI Model. Where existing protocols are not incorporated, interfaces between them and the OSI Model are being implemented. This is the case for TCP/IP, NetBIOS and APPC, the major middle-layer protocols available today.

In the PC LAN environment, NetBIOS is the most important protocol. It stands for Network Basic Input/Output System. IBM developed it as a BIOS for networks. It is essentially a Session layer (Layer 5) protocol that acts as an applications interface to the network. It provides the tools for a program to establish a session with another program over the network. Hundreds of programs have been written to this interface, making it the most widespread protocol in the PC network arena.

NetBIOS does not obey the rules of the OSI Model in that it does not talk only to the layers above and below it. As we said, programs can talk directly to NetBIOS, skipping the application and presentation layers. This doesn't keep NetBIOS from doing its job. It just makes it incompatible with the OSI Model, which is not the end of the world. Someone will write an interface between the two, soon.

NetBIOS is limited to working on one network. Therefore, some network vendors have established an interface between NetBIOS and TCP/IP, a protocol from the Department of Defense for use over large combinations of networks (internetworks).

TCP/IP stands for Transmission Control Protocol/Internet Protocol. TCP is a Transport protocol (Layer 4), corresponding to the definition we gave above. Its job is to get data from one place to another without errors. It forms an interface between the protocols above and below -- shielding the upper layers from concern about the connection and the lower layers from concern about transmission content.

The IP protocol is for getting data from one network to another. Its main concern is bridging the differences between networks so they don't have to be modified to talk to each other. It does this by providing rules for the breakdown of data to conform with a given network. Gateways, which are the physical translators between networks, use IP's rules to take data from one network, modify it and route it correctly over another network.

TCP/IP enjoys enormous support in government, scientific and academic internetworks. These computers use UNIX and other large-computer operating systems. In the past few years, business internetworks have begun to approach the size of those in government and universities. This has driven these businesses to look for internetwork protocol standards. They have found TCP/IP useful and it has become a de facto standard. Many see it as an interim solution until the OSI transport and internetwork protocols are finished. TCP/IP products for DOS-based networked PCs are also available.
Often when TCP/IP is discussed, acronyms like SMTP, FTP and TELNET are tossed around. These are applications that have been written for TCP/IP and are widely used. They work at the Applications layer (Layer 7). SMTP stands for Simple Mail Transfer Protocol. FTP stands for File Transfer Protocol. TELNET is the name for a terminal emulation protocol. These protocols, written for TCP/IP, do exactly what they say they do.

Advanced Program-to-Program Communications, or APPC, is another protocol for large networks. It comes from IBM and is part of Big Blue's Systems Network Architecture (SNA). It is similar to NetBIOS in that it provides an interface to the network for programs so they may communicate, but it is not limited to one network as is NetBIOS. APPC is geared toward mainframe computers, though IBM is offering it as part of its OS/2 Extended Edition. Using APPC, all computers communicate as peers, even PCs. Previously in the IBM world, PCs were forced to emulate terminals when communicating with mainframes. A number of other vendors, mini and micro, also offer APPC.
APPC has received much publicity. Unfortunately, there are not many applications for APPC in the PC network arena. There are more in the minicomputer and mainframe network market. Nevertheless, IBM and others are promoting APPC as a protocol standard for the future. Its robustness, flexibility and reliability make it worth the extra development effort.

There are other middle-layer protocols. XNS, IPX and NetBUEI are all transport protocols. XNS is short for Xerox Network System. It was one of the first local area network protocols used on a wide basis, mainly for Ethernet (802.3) networks. 3Com and many others use it. IPX is Novell's implementation of XNS. It is not completely compatible with the original, but very widely used. NetBUEI is IBM's transport protocol for its PC networking products. All of these protocols perform similar tasks.

Many More If it seems like the number of protocols is idiotic, it is and it isn't. Different protocols have different advantages in different environments. No single protocol stack will work better than every other in every setting. NetBIOS seems to work fantastically in small PC networks but is practically useless for communicating with mainframes. APPC works well in mainframe environments. TCP/IP excels in large internetworks.

On the other hand, much more is made about the differences in protocols than is actually warranted. Proprietary protocols are perfect solutions in many cases. Besides, if the proprietary protocols are widespread enough, they become standards, and gateways between them and other standards are built. This is happening with some of the major protocols we have not covered. These protocols include many de facto standards in minicomputer and scientific workstation communications. They include DEC's entire protocol suite, Sun Microsystems' NFS, AT&T's protocols and many others. We have also left out Apple's AppleTalk and AFP. While these enjoy widespread use, that use is based on the computers these companies are selling and not the proliferation of the protocols throughout the networking industry.

Unfortunately, whether proprietary or standard, users are still faced with the dilemma of choice. This choice is made slightly easier by the shakeout and standardization that has occurred over the past few years at the lower Physical and Data Link layers. There are three choices, Token Ring, Ethernet or Arcnet. Right now, the same is happening at the higher layers. Can you guess which way things will go?

*** * ***

BEGINNERS' CORNER

At 7:00 p.m. prior to the regular monthly OPCUG meetings, sessions are held for beginners, where you can ask questions and discuss problems.

If you have a special topic you would like to have discussed at one of these sessions, phone Eric Clyde at 749-2387.

OTTAWA PC USERS' GROUP - EXECUTIVE & ASSISTANTS

Chairman	Douglas Poulter	745-8768
Past Chairman	David Terroux	238-4895
Treasurer	Tony Frith	671-0401
Secretary	Norman Dafoe	723-1909
Newsletter Editor	Bonnie Carter	236-1015
Software Librarian	Chris Taylor	723-1329
Membership Chairman	Harald Freise	828-3411
Convenor	Paul Green	820-5348
BBS System Operator	Jean Fortier	236-1015
Hardware/Software Broker	Terry Mahoney	225-2630
Software Assistant	John Ings	235-8132
BEGINNERS' SESSIONS	Eric Clyde	749-2387

Please Print **MEMBERSHIP APPLICATION (renewal)**

Last Name:		First Name:
Address:		
City:		Province:
Postal Code		Country:
Telephone: Home:_____ Office: _____ Fax:_____		
Are You: _ A New Member? _ Renewing your membership? I.D.#_____	Sponsor's Name:	MEMBERSHIP FEE $25.00
Do you wish to subscribe to the Disk of the Month? (10 disks per year)	_ Yes _ No	Format: _ 5.25" @ $25.00 yr. **$.** _ 3.50" @ $35.00 yr. **$.**
		Total: _ Cheque _ Cash **$.**

Can you help in Group Activities? Check those that apply.

_ Programming Instruction	_ Hardware Techniques	Hardware Used:	Modem?
_ Newsletter Input	_ Meeting Locations		_ Yes
_ Memberships	_ Agendas & Speakers	_ XT	Baud:
_ Software Library	_ Advertising	_ AT - 286	_ 300
_ Promotion/Publicity	_ Bulletin Board	_ 386	_ 1200
	_ Other _____	_ 486	_ 2400
			_ 9600

What in particular interests you in the Group?