AMIGA

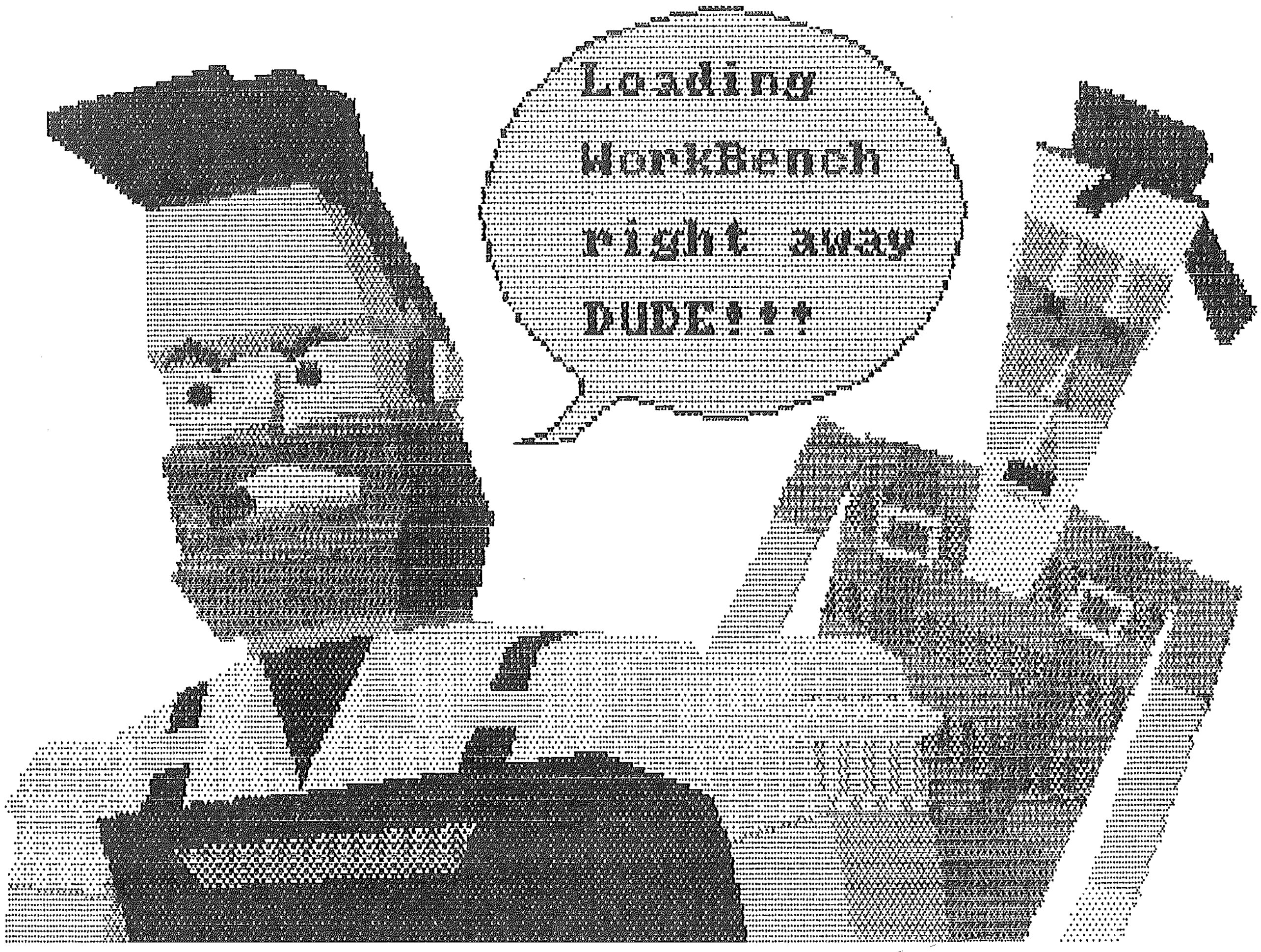
$2 RRP

WORKBENCH

FOR THE COMMODORE AMIGA USER

Registered by Australia Post – Publication No. VBG7930

Number 8 Circulation: 450 January 1987



Next Meeting

No formal January meeting. Instead, come to our Barbecue on Saturday, January 17th from 12 noon onwards

The January Barbecue will be held at the Studley Park Boat Sheds – Melways map 44 reference H4. Come along!

Amiga User's Group, PO Box 109, North Balwyn, 3104, Victoria, Australia

AMIGA™ Users Group

P.O. Box 109, North Balwyn, Victoria, 3104

Amiga Users Group

The **Amiga Users Group** is a non-profit, self-help group, made up of people interested in the Amiga computer and related topics.

Club Meetings

Club meetings are normally held at 2pm on the second Sunday of each month at Victoria College, Burwood Campus, in Lecture Theatre 2.

In January, however, we will be having a barbecue at Studley Park Boat Sheds, off Studley Park Road, on the border of Kew and Collingwood, Melways map 44, reference H4. Since some of our members might be still on holidays on our normal meeting date, and since some of our members belong to another computer club, Micom, who are having a barbecue a few days later, we have changed the date to the **third Saturday**, that is **January 17th, 1987**, from about 12 midday onwards. We will supply the soft drinks, and money for the barbecues, all you'll need is the food and eating utensils.

Production Credits

This month's **Amiga Workbench** was edited by Peter Jetson. Equipment and software used was: TurboDOS S-100 computer, Diablo 630 printer, Gemini 10x printer, Wordstar and Fancy Font. Now that Desktop Publishing software for the Amiga is becoming available, we might soon be able to do the whole newsletter on an Amiga.

Copyright and Reprint Privileges

Contributions

Articles, papers, letters, drawings and cartoons are actively sought for publication in **Amiga Workbench**. It would be appreciated if contributions were submitted on disk, since that means they don't have to be re-typed! We have access to a wide range of computers, so we should be able to accept almost any type of disk, but Amiga disks are certainly the easiest. Absolute deadline for articles is the last weekend of the month before the cover date. Contributions can be sent to:

The Editor, AUG, PO Box 109, North Balwyn, 3104

Membership and Subscriptions

Membership of the **Amiga Users Group** is available for an annual fee of $20. To become a member of AUG, fill in the membership form in this issue (or a photocopy of it), and send it with a cheque for $20 to:

Amiga Users Group, PO Box 109, North Balwyn, 3104

Amiga Users Group Committee

John Holland	(Co-ordinator)	348 1358	Carlton
Bob Scarfe	(Vice Co-ordinator)	376 4143	Kensington
Ron Wail	(Meeting chairman)	878 8428	Blackburn
Eric Salter	(Secretary)	861 9117	Kew
Paul Radford	(Treasurer)	663 3951	BH only
Neil Murray	(Membership)	792 9666	Dandenong
Stephen Thomas	(Membership)	830 5783	Canterbury
Peter Jetson	(Newsletter editor)	762 1386	Boronia
Geoff Sheil	(Assoc editor)	509 3151	Armadale
Jo Santamaria	(Assoc editor)	836 9129	Canterbury
Ron Van Schyndel	(Librarian- books)	882 7264	Hawthorn
Geoff Wood	(Librarian- books)	580 7463	Aspendale
Ron Wail	(Librarian- software)	878 8428	Blackburn
Mike Creek	(Librarian- software)	878 9039	Blackburn
Bohdan Ferens	(Purchasing)	792 1138	Dandenong
Justin Somers	(Without portfolio)	553 0379	Moorabbin

When phoning committee members, please try to be a bit considerate and not call at meal-times, late at night, or during popular TV programs. If you only have a general query, try to ring the member who lives closest to you.

Member's Discounts

The **Amiga Users Group** is currently negotiating discounts for its members on hardware, software and books. Members will be notified when negotiations are complete.

Currently, **Technical Books** in Swanston Street in the city offers **AUG** members a 10% discount on computer related books, as does **McGills** in Elizabeth Street. Just show your membership card. Although we have no formal arrangements with other companies yet, most seem willing to offer a discount to AUG members. It always pays to ask!

AUG Users Group Disks

Disks from the **Amiga Users Group** Library are available on quality 3.5" disks for $10 each including postage on AUG supplied disks, or $2 each on your own disks. The group currently holds 47 public domain volumes, sourced from the USA, with more on the way each month.

Making It With Make

For the Lazy and Ingenious

by John Toebes and Jack Rouse
The Software Distillery
235 Trillingham Lane
Cary, NC 27511, USA

Make is a UN*X utility which allows you to keep track of your source files no matter how large or complicated your project gets. Suppose you have two source files for a C program: main.c and subs.c. Let's also say they both #include foofile.h. If you change main.c, you must recompile main.c and relink. If you change foofile.h, you must re-both main.c and subs.c. This isn't too bad with only two files, but just try to keep track of, say, eight source files...ten...a hundred. **Make** is designed to help you whether you have one, ten, or one hundred source files.

The principle behind **Make** is that you specify all the commands required to produce your program in a file (called 'makefile'). Example:

```
program: main.o subs.o
blink from main.o,subs.o to program
echo "all done!"
```

When you want to produce an up-to-date version of your program, you simply type 'make'; **Make** looks at the system dates on your source files, dedermines which must be recompiled, then recompiles them and relinks your executable! A makefile for our simple program with two source files is shown below:

```
main.o: main.c
foofile.h
lc main.c
subs.o: subs.c foofile.h
lc subs.o
```

This simple script tells **Make** that the file PROGRAM depends (:) on the files main.o and subs.o, and that to make the file PROGRAM you execute the 'blink' command on the second line. You can have as many filenames separated by blanks as you wish after the colon. Commands, if present, must start with a blank or tab. You may have as many commands as you wish.

Note that there are entries for main.o, subs.o, and foofile.h. **Make** will see that PROGRAM depends on main.o, and main.o depends on main.c; it will check the dates on main.o and main.c to see if the 'lc main.c' (compile) command should be performed before it links PROGRAM.

Instead of entering an lc command for each .c file you have, you can instead enter a special dependency that looks like the one below:

```
c.o:
lc $*.c
```

This tells **Make** that whenever it sees an .o file, it must see a dependency upon a .c file with the same prefixed name. You are relieved of any need to put this dependency in every makefile; instead, **Make** allows you to put it in a file called "builtins.make" in your C: directory. If the expression "$*" is seen in a builtin dependency, **Make** replaces it with the file name being processed (minus the .c or .o extension, of course).

Now that your makefile is entered, how do you use it to make PROGRAM? Quite simple. Just type 'make'. **Make** looks in the makefile and sees that PROGRAM is the first file listed; it therefore assumes that you mean to make PROGRAM. It then does whatever is necessary to bring PROGRAM up to date. But what if you just want to compile 'main.c'? Also simple. Type 'make main.o'. **Make** takes the argument and figures out how to make it from the makefile.

In fact, if you have typed in the .c.o dependency above, you can **Make** FARKLE.O when FARKLE is not even mentioned in the makefile. As you can see, **Make** is worth its weight in saved keystrokes alone!

There are several useful command line options on **Make**: "make -n ..." prints the commands needed to bring "file" up to date to the screen but will not execute them. This is great if you just want to see how out-of-date your stuff is. A "make -k ..." tells **Make** that it should proceed as far as it can without stopping. If a command fails, **Make** quits without this flag; with the flag, **Make** continues until it needs the output file from the failed command.

This means that if you compile 50 files and number 27 fails, numbers 28 to 50 are compiled before **Make** needs the .o file for number 27. Last, "make -f <filename>" tells **Make** to get its instructions from <filename>, not from "makefile."

Make observes ^C interrupts; hit ^C and everything stops. It also observes ^D interrupts; hit ^D and **Make** quits after the current command is executed.<

For the Advanced User

There is a limited ability to define macros in a makefile. Macros are of the form NAME = STRING. Whenever the string $(NAME) is seen in a makefile, STRING is put in its place. For example, you can use the macros below:

```
LC = lc $*
LCD = lc -d $*
.c.o:
$(LC)
```

This allows you to change your default compiler options rapidly by simply changing the last line to read $(LCD) [with a -d option] in place of $(LC).

Normally, **Make** types each line to the screen before executing it. If you prefer silent execution, preface a command with an '@'. Example:

```
program: program.o
@echo "Linking program. . ."
@blink from lib:c.o,program.o to program lib lib:lc.lib
```

Enjoy **Make** as much as I do! Many thanks to John Toebes and Jack Rouse for **Make**ing it available!

[Editor's note: This text file is taken from the public domain disk **Amigan #4**, which also contains the **Make** program.]

Happy New Year from AUG!

An Introduction to Writing Your Own Device Drivers

by Alan Kent

Ron Wail and co. are not the only ones who have been working on a hard disk for the Amiga. I have also been working on one for some time now in the little spare time I have. As I am currently doing my masters in computer science at RMIT, I do not have as much time as I would like to spend on the machine. If you wish to **buy** a hard disk you will have to buy one from Ron and friends as I have no intention of building and selling them. However, if you want to take on the challenge of building your own, then read on!

Getting the additional hardware needed to get a hard disk to work was trivial compared to the problems I had trying to get the software to work (this may be because my brother John designed the hardware for me). I used a WD-1002-05 controller card as my brother had one lying around at home. I recently rang Daneva Australia in Sandringham and they said the boards now cost approximately $550 plus 20% tax (single quantities). Perhaps there are cheaper boards around these days. The controller card can in fact handle 3 hard disks and 4 five and a quarter inch floppy drives so you can also add some standard 5 inch floppies too if you like.

Using a prebuilt controller card made the hardware quite simple - in fact, only 7 additional IC's were needed for decoding and timing. At present, I have the IC's on a wire wrap board which plugs into the expansion slot on the side of the Amiga. Someday, I hope to add a bit more memory and a battery backup clock. I will explain the hardware in more detail later.

In this article, I thought I would explain some of the fundamentals of trying to write a device driver. It may not be enough for you to write your own driver immediately, but its a step in the right direction. Much of the information here is relevant for any device driver, not just a hard disk device driver. All of the information in this article came from the manuals (somewhere), mainly the two volumes of the RKM (Rom Kernal Manual). Much of the hardware information I used came from the expansion specs which I ordered from Commordore in the states for US$40(?). As I am a C programmer by nature and dislike having to write assembly language code (although it is faster), I have added an extra twist to this project of trying to implement my drivers in C. I think I should point out at this stage that it is almost impossible to add hard disks to V1.1 (I tried for weeks, but I just could not get AmigaDOS to realise that the hard disk existed). It is very easy however with V1.2.

EXEC and AmigaDOS

Before we can really get into device drivers, some basics of the multitasking executive (called Exec) must be known. Exec is the low level program that has a number of responsibilities, the main responsibility being to decide which task is to run next on the Amiga. A task is simply an occurence of a program running. For example, if two CLI windows are open at the same time, then there are two CLI tasks running. Each task on the Amiga is given its own memory space and stack space. Details about the task, such as where its stack space is and the contents of the task's registers when the task is not running, are kept in a task structure. Exec also provides mechanisisms for supporting communication between tasks, support for libraries of functions and support for devices, as well as keeping track of what memory has been allocated.

AmigaDOS is not part of Exec. It is another level of software which runs above Exec. AmigaDOS is basically a file management system which uses the Exec functions. Its job is to keep track of where files reside on the disk and provide routines to read from and write to files. When AmigaDOS starts running a program, it actually starts a task, but adds some more information to the end of the task structure such where as to direct console input and output. Such tasks are called processes. Another slight difference is that a pointer to a process points to the first field in AmigaDOS's extra information after the task structure. The first entry happens to contain a pointer to a message port which AmigaDOS uses to communicate with the process (more on message ports later). Care must be taken to determine if a pointer is a pointer to a process or a task. In C, a process pointer can easily be converted to a task pointer using the following code:

```
task = (struct Task *) (((char *)process) - sizeof(struct Task
```

Signals

For one task to communicate to another task, Exec provides a number of functions. Each task keeps 32 signals. These signals can be set by other tasks using the Signal() function. The task receiving the signal can then use the Wait() function to wait for a signal to arrive. When the Wait() function is called, Exec will remove the task from the list of running tasks until the desired signal arrives. This means that no CPU time is used by the waiting process allowing more CPU time for other processes. A task can actually wait for any one of many different signals to arrive from many different tasks at the same time. Signals thus provide a very simple synchronisation method between tasks. One problem with signals, however, is that if the same signal is sent twice to a task before the receiving task gets a chance to have a look, it appears as if only one signal is received.

Messages

Signals by themselves are not particularly useful. What is more useful is to be able to send some information from one task to another. Messages and Ports provide this ability. A message is simply a block of memory which has a special structure at the beginning of the block followed by any other information that is to be sent. In C, this can be achieved by defining a structure as follows:

```
    struct my_message {
                /* first the standard message information */
        struct Message msg;
                /* and now my details */
        int number;
        char character;
        int other_information;
        char *string;
    };
```

The message structure definition can be found in the include file <exec/ports.h>. The Message structure has to be set up in a special way which is described in the RKM Exec manual, page 34. Signals are used to notify a task that a new message has been received.

Ports

Ok, now we have a message structure, but how do we get a message from one task to another? And how does one task know which of the 32 signals to send to the other task to notify it that a message has been sent? This is where ports become useful. A port is like putting a letter box outside your house. It provides a known place that messages can be received. A port can be created using the function CreatePort() and when created, can be assigned a unique name. The function FindPort() can be used to search the whole system for a port by its name. When a port is created, it is also allocated a signal number which is used to tell the receiving task that a message has arrived.

Sending Messages

To send a message from one task to another then involves using the PutMsg() function. PutMsg() requires a pointer to the port (as returned by FindPort) and a pointer to the message to be sent. The actual sending of the message involves letting the receiving message port know where the message is in memory and then sending the receiving task a signal to let it know that a message has arrived. In order to allow many messages to be sent to a single port, messages are kept in a queue. This is why some special information is needed at the beginning of each message structure - list pointers must be maintained.

The receiving task normally waits for messages to arrive using the WaitPort() function. This basically involves doing a Wait() for the signal bit assigned to that port. Once a message is received (or if a message has already been received and the program had not noticed it yet) WaitPort() will exit. Messages can then be removed

from the port using the GetMsg() function. Note that due to the problem mentioned before about signals and the possibility of multiple signals appearing to be only a single signal, many calls to GetMsg() may be required. It is also possible that WaitPort() will return that a new message has arrived but it had already been processed - its just that the signal bit had not yet been cleared. The normal code structure for receiving messages can be safely performed using the following C code:

```
    struct Message *msg;

    while ( 1 ) {        /* loop forever */
        WaitPort ( message_port );
        while ( ( msg = GetMsg ( message_port ) ) != NULL ) {

            /* do something with the message! */

        }
    }
```

Until a message has been received and processed, the sending task should not modify the contents of the message. The PutMsg() call is basically granting the receiving task permission to use that piece of memory. Once the receiving task has finished with the message however, it may be useful to let the sending task know. This is in fact very important if information is to be returned to the sending task in the message structure. What is done then is for the sending task to also create a message port. This message port however is not given a name and so cannot be found using FindPort(). Instead, a pointer to this port is kept in the actual message structure (in the field mn_ReplyPort). The task that receives the message then uses the ReplyMsg() function to effectively send the message back to the original task. This means the the sending task must do a WaitPort() on the reply port so that it will wait until the message is sent back. This allows complete synchronization between the two tasks. Note that if the reply port is not put into the message structure, then ReplyMsg() does nothing.

To close off a port so that no more messages can be received, use DeletePort(). The functions CreatePort() and DeletePort() are actually support functions written in C which should be in the libraries that come with your C compiler. These functions allocate/deallocate the necessary memory and call (if necessary) the functions AddPort() and RemPort(). For more details on ports and some example code using them, see chapter 3 in RKM: Exec.

What is a Device Driver?

Well, now let's get into what this article is really about! A device driver is a piece of code which provides an interface between the Amiga operating system or user programs and (usually) hardware on the system. There is one device driver for controlling the floppy disks, another for controlling the printer and so on. Drivers usually reside on disk until such time as they are opened. When a device is opened, the Amiga searches through its list of known devices in memory and if it is not found, it then searches the devs directory on disk. This is why the first time you use the printer, the disk will become active for a period of time (to load the device driver), but when using the printer later, there may not be any extra disk activity (as the driver is already loaded). Note that the Amiga can try to reclaim memory from a device driver if it is not in use and the Amiga is low on free memory.

Using C it is not easy to get this auto-load feature to work. As a result, I decided to ignore it by instead adding a RUN command in my startup-sequence file so that the driver was permanently loaded and could not be removed. This may not be very elegant, but it works!

In order for a program to make a request to a device driver (for example the printer driver), an OpenDevice() call must be made. This call has 4 parameters: the name of the device you wish to open, the unit number you want (some devices such as the floppy disks can have several units - for efficency they all share the same device driver code), an IO request block and some flags which are interpreted by the device driver. The IO request block is a message with some extra information added (see struct IORequest in <exec/io.h>). This extra information includes a pointer to the device, a pointer to special memory allocated per unit, a command field, an error field and a flags field. As many devices perform much the same sort of commands (for example, writing to disk is not that much different than writing to a printer) a standard form of requests has been defined. The structure and basic commands defined are also in <exec/io.h>.

Once a device is open, requests can be made using the IO request block passed to the OpenDevice() call using the functions CheckIO(), DoIO(), SendIO() and WaitIO(). DoIO() performs an IO operation by sending the device driver the IO request as a message and waiting for the driver to complete (signaled by a ReplyMsg()). To be more precise, DoIO() tries to call the BeginIO entry point in the device driver directly (explained later) which will either immediately process the command and return, or else send the message to the port for processing when the driver is ready for it. SendIO() sends the request to the device driver's port, but does not wait for completion. This allows the task to continue doing other things. WaitIO() can then be used to wait for the command to complete. CheckIO() provides a means of testing if the operation is complete without waiting for it if it has not.

When CheckIO() says the operation is complete, WaitIO() must still be called to clear the signal bit. The RKM: libraries and devices contains many pieces of example code opening devices, sending commands and closing devices. Note that it is very important to close a device when finished with it as some of the devices only allow one task to open them at a time.

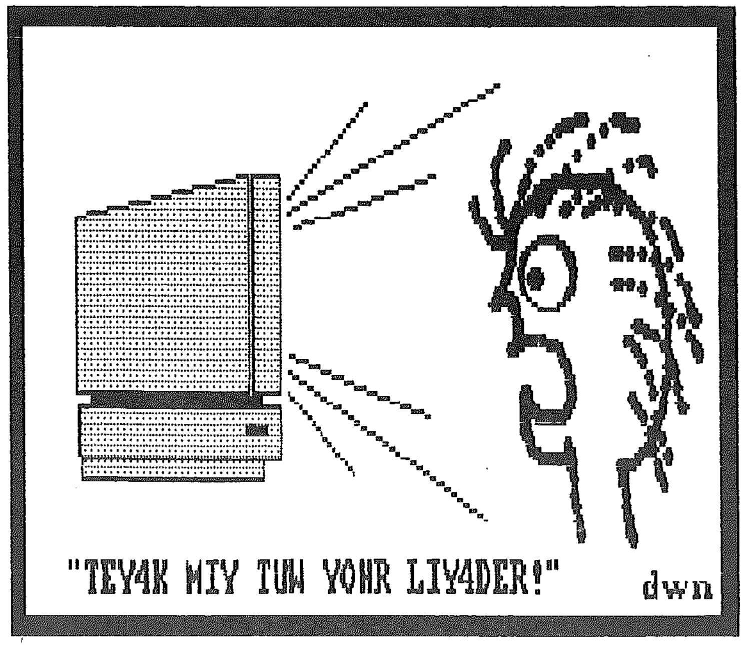
[In part 2 of this article, to be presented next month, Alan will continue with information about writing a hard disk device driver, and will present details on the hardware used in the project.]

Chuck Another Prawn On The Barbie!

You **are** coming to the first annual Amiga Users Group Barbecue, aren't you? Of course you are! How could you miss out on the social event of the year!

We've planned our BBQ to coincide with the one that Micom (The Microcomputer Club of Melbourne) are holding, so that our many members who belong to both groups don't come down with BBQ overload. So, we're holding it a few days later than our normal second Sunday of the month. If you haven't already circled January 17th on your diary, **do it now!**

The place to be on that date, with your family and/or friends, is near the Studley Park Boat Sheds, in Kew. Find it on Melways map 44, reference H4. We'll kick off about 12 noon, with AUG supplying softdrinks and coins for the BBQs. You'll need to bring your own food, non-soft drinks, plates and utensils.



DBMAN

Preamble

Some time ago, I began designing a business system which was to be my first attempt at using the Amiga for anything other than games. I soon realised that even the simple system I had in mind was going to take months to write in Basic (and even longer if I had to learn another language). So, I decided to investigate Amiga database packages.

Trying to evaluate business software is difficult enough when you know what you are doing, but in this case I had only a vague idea about what to expect, and I found that very few shopkeepers really know their products. Maxwells were kind enough to let me clutter up their showroom for several days while I tried to educate myself in databases and evaluate some of the available packages.

What is a database?

A database is a collection of files, usually with some sort of indexing system. A non-programmable database package allows you to define the attributes of a file, sort the file over one or more keys, add new records, delete old records and change the attributes of existing records. The better packages can do this over several files at once, and even update one file from another. This means that once you have data in a file, you never need key it in again, even if you require it in a different format in a different file.

A programmable database, in addition to these features, has a Basic-like language which contains all the commands found in the non-programmable package.

What is DBMAN?

DBMAN (based on the famous dBASE II) is one of the few programmable database packages available for the Amiga. Unfortunately, because DBMAN was not written from the start for the Amiga (ie it is available for other computers), it does not use all of the Intuition interface bells and whistles that the Amiga has available. On the plus side, however, it doesn't have the bugs or design flaws found in many new products.

DBMAN may be used in a programmable or non-programmable mode. The documentation is thorough (about 250 pages), includes exercises for beginners, and there are sample programs included on the disk. It does assume familiarity with Ed in CLI if you wish to use the programmable portion. As a language, the programmable section is easier to use than Basic, but some of the functions work in unexpected ways, so there are a few traps to be wary of. The DBMAN disk has no copy protection, and once DBMAN is loaded, the disk may be removed to free the drive for data disks.

Performance

In most areas, the performance exceeds dBASE III, which is widely regarded as the standard for this type of package. The interpreter runs at over twice the speed of dBASE III on the IBM and it can support more open files. The file sizes can be bigger than even a hard disk will allow, and access speed seems adequate. However, two areas cause some concern. The first is the time required for a file pack. This is the function of purging records that have been marked for deletion. It seems that no buffering is used by this process, so it is possible to give a pack command and have to wait three hours for it to finsh! The solution to this is to save your pack until you are ready to watch the TV. The other area of concern is that if there is a disk read error or some similar problem, you are locked out until you use the Amiga N to get behind the screen to the alert or requester window. Before I became aware of this trick, I had to reboot each time this sort of thing happened.

Conclusion

At half the price of the IBM product, DBMAN equals or exceeds it in most areas. DBMAN can do in minutes what it might take a Basic programmer hours or days to write.

-- C. Stinson

Mind Walker: A review by Stanley Thomas

You are a physics professor who, thinking too much about the strange world of quantam mechanics, has gone insane. Keen to get your sanity back, you go on a journey inside your own head. The first thing you need to do is travel through your Mind and trace a single coherent thought; not an easy business since you are constantly attacked by Bad Thoughts beaming Nihilism Beams, and Existential Deathmasks. To do this you must traverse a landscape consisting of a network of Blocks and Platforms armed only with a fractal ray, using any one of four character types consisting of a Human, a Wizard, a Spriggan or a Nymph. When you have done this, you then descend into the Brain, where you must search a maze of neurons looking for seven Shards of Sanity.

This done, you take your precious Shards down to the last part of your Journey, the Subconscious, where you have to position the Shards in an Inkblot. To help you do this, Uncle Sigmund is on hand to help you. Accomplish this, and you have completed the first of seven journeys, each one harder (of course) than the last.

This very bare description does no justice to the excellence of Mind Walker, Commodore's one and only game for the Amiga. It does superbly what I think any good computer game should do- it uses all the elements of the game to create a world with a distinct mood and feel to it, a world you can enter and become involved in for the duration of the game, and leave having been challenged, amused and entertained. More than anything, it is very compulsive game. Even though I had to start work early the next day, I found myself playing this game until well past midnight!

Mind Walker uses the graphics capability of the Amiga to very good effect. The Bad Thoughts which attack you are fully rounded and make fine use of subtle shading. The landscape of the Mind is a barren and strange place, with its three-dimensional network of possible pathways. The screen representing the Brain is more conventional, being as it is a maze, but it too makes good use of colour and graphics effects. The last screen, the Subconscious, uses colour cycling in the style of Deluxe Paint and Polyscope. Dying is also a quite spectacular event and well worth a few suicides to see the graphics.

Sound is one of the great strengths of this game, and perfectly match the stage of the game you have currently reached. The music which accompanies your journeys around the Mind flickers with intriguing melodies and harmonies. The descent to the Brain is full of sighs, and the Brain has its own strange sounds, as does the subconscious, where the music is darker and more sombre, well befitting such a mysterious realm.

Unlike Marble Madness, you don't have to start from the beginning again every time. You are given two options. Firstly, you can save a game at any stage, allowing you to continue later. Secondly, you can set the difficulty level by changing a sliding control on the menu bar, which has the effect of starting you off at different stages of the game.

When I owned a Commodore 64, the upper price limit for a game was about $80. For the Amiga this is the starting price, forcing me to be much more choosey about what I buy. The price for Mind Walker is an expensive $129, but I consider it money well spent for what is one of the best games I have ever seen. An added bonus is that the License Agreement permits you to make a backup copy of the disk, which is unprotected to allow you to do this. Personally, with software for the Amiga costing as much as it does, I feel that we have the right to make backup copies for our own personal use, especially as disks seem to wear out. My copy of Deluxe Paint is taking longer and longer to load, with the drive making all kinds of disturbing grinding noises as it tries to read the program from the disk. It hasn't failed yet, but I'm waiting for the day it finally does and I have to start using the second copy of the two provided. I'm glad the trend seems to be moving away from protection schemes that, in their bid to stop people copying their work, become increasingly sophisticated and fussy about the condition of the disk and the alignment of the drive, and increase the load time of programs and make life difficult for owners of hard disks. Goodonya, Commodore!

Customising Your S/Startup-Sequence

Have you ever wished that your Amiga could boot up with a happy smiling face, just like the Mac? Probably not, but you can easily set up your Amiga to say, do or look exactly as you wish. I have made several different variations of startup-sequence, and had lots of fun doing it (and showing it off).

But before we get into changing the startup sequence, I would recommend you get a handy little text editor called TED, this is available on public domain Fred Fish disk #20. I use TED to create my startup sequence and many for other projects and found it to be very helpful, I think you will find it useful.

Before you do anything mentioned in this article, I STRONGLY suggest that you make a backup of your Workbench disk to experiment with. You can alter your startup-sequence to display anything from a simple one line message to a full screen picture. First try opening the directory "S" and display the file "s/startup-sequence". Your startup sequence will look like this in its unaltered form;

```
echo "workbench disk.  release 1.1"
echo " "
echo "use preferences tool to set date"
echo " "
loadwb
endcli > nil:
```

This is realy not unlike BASIC, where echo is the same as BASIC's PRINT (writeln, printf or any other language you wish to compare it to). All text is enclosed in double quotes ("these things"). To make it easier to read, it is a good idea to use blank lines (echo " ") in selected places, which is just like a blank PRINT statement.

After making a backup, I would suggest that you experiment with the startup-sequence to see what you can achieve. You can't do any real damage, and experimenting is a good way to learn. You can start by just changing one line, for example, by replacing the first line with this:

```
echo "Welcome to workbench, master"
```

Now, resave "s/startup-sequence", then reboot to see your new look, grovelling workbench.

I have found that the best effect comes when a full screen picture is displayed (The picture on the cover is from my startup sequence). To do this, you will need the SHOWILBM command from the public domain Fred Fisk disk #12. Copy the SHOWLIBM command to the workbench disk, then find a picture to use. This can be any IFF picture that takes your fancy, there are many on the PD discs (but it MUST be IFF) or you can draw your own with dpaint, images or whatever.

Anyone familiar with MS-DOS will see that the startup sequence in AMIGA-DOS is very similar to the AUTOEXEC.BAT in MS-DOS. The startup-sequence file can be used for many different things, besides just displaying pictures. You may want to make your own auto-boot slide show. To do this, make a copy of Work Bench, delete all the unnecessary files, (fonts DIR, empty DIR, utilities DIR and many others can go), and include DPSLIDE DPSLIDE.CMD in the startup-sequence (but delete loadbw). Copy dpslide and the pictures from the dpslide disk (Fred Fish disk #12) onto your workbench disk.

I hope this has helped you to understand what the "S" directory is for and what it can do. Please continue to experiment with things like the startup-sequence; it is still early days for the AMIGA and there are many things it can do that are just waiting to be discovered.

Most of all, I hope you enjoy as you learn about this "AMAZING COMPUTER".

-- Fergus Bailey

Stuff You Might Like To Know

Happy New Year! The Amiga Users Group welcomes you to 1987. It's raining as I write this, perfect weather to stay indoors with your Amiga.

It seems that the official **Version 1.2 AmigaDOS package** was released in the USA just prior to Christmas. The **Amiga Enhancer** package costs $14.95 (in US dollars!), and includes version 1.2 of the Amiga Operating System on two disks (Kickstart and Workbench), a third disk with a revised Microsoft AmigaBASIC, and some documentation. Contrary to what has been rumoured in **Amazing Computing** magazine and other places, V1.2 **does not** come with a utility to read and write IBM format disks on an external 5 1/4 inch drive - it does, however, allow you to use a 5 1/4 drive as a limited capacity AmigaDOS drive.

We still have no news on when the upgrade package will become available from Commodore locally.

At about the same time that the Americans were getting the V1.2 upgrade, we were finally getting the **SideCar**. Unfortunately, not as cheaply as they'd been telling us - the final retail price turned out to be $1299, not $999 as promised. At that price, you could by a PC clone and have two computers. Strangely enough, Sidecar comes with AmigaDOS V1.2!

According to **The National Amiga Users Group** Newsletter (from the USA, dated 01-Dec-86), it appears that Commodore only sold **35,000 Amigas** between September 1985 and September 1986. They note that estimates had always been about 40% higher. Also noted is that **Atari** sold only **75,000 520ST and 1040ST** computers during the past 18 months, and that estimates had always been 100% higher than that.

An interesting advert in the latest issue of **Amazing Computing**: Digi-Pix of California will digitise your pictures from flat art, 2" x 3" to 8 1/2" x 11" or color slides from 35mm to 4" x 5" using the Digi-View system to 32-color, 320 x 200 resolution pictures, compatible with any IFF paint program. Seems to me that with the high local price of a Digi-view system and camera, an enterprising hobbyist could start a similar service here and make some money with their Amiga.

A month or two back, the Amiga Users Group purchased a **Future Sound Audio Digitiser** for members to borrow. According to Paul Radford, who is handling the bookings, the project has so far been a great success. The unit is in great demand, with about 16 people on the waiting list at the present time. With a two week loan period, unfortunately that means we're booked up for about **8 months!**

Quite a number of people have now upgraded (?) their Amigas with **MC68010** processors. The latest issue of **The Amigan Apprentice & Journeyman** contains a how-to article on the upgrade. Their benchmark tests confirm our results - improvements of between zero and 5%. A small minority of programs **do** exhibit drastic speed improvements - Mandelbrots are up to 25% faster. The general feeling, however, is not to bother unless you really feel the need to say "Mine is a little faster than yours!".

Also in **A & J** is a nice note about us. I sent them a newsletter so we could be included in their **User Groups** column. Amongst other things, they said "If we lived down under, we'd join these folks if our house was in Alice Springs or 500 miles in the outback." The best compliment, however, was that they asked permission to reprint several of our articles, and a request that we interchange Journals.

In my article on printer drivers in the October newsletter, I said that the **BLINK** I had couldn't be convinced to make printer drivers, and that I had to return to using ALINK. Well, by a lucky chance (I'd call it serendipitous if I could spell it!), the folk at **A & J** are friendly with the people at the **Software Distillery**, the makers of BLINK, and they sent us the very latest version, 6.7. I'm happy to report that BLINK now links printer drivers that do not send for the guru. Thanks, guys.

In the public domain update section of the newsletter, I mentioned that the Amicus disks still hadn't turned up. Well, I just called **Amazing Computing**, and they sent them a few days after Christmas, which means we should have them by the BBQ. We ordered the disks in late October, and since then there have been 3 more releases. I think we'll wait until the first lot show up before we send any more money!

I've been in two minds over whether to make this next item public. Some people have suggested we might be on a better footing with Commodore if we don't talk about their shortcomings. I'm still not sure, but here it is.

You may be aware that I am a registered Amiga developer. That means I am supposed to get information about the Amiga from Commodore that will help me write programs and develop hardware for the Amiga. So, what have I got so far? You may remember the tales of woe I've told in past columns, about sending heaps of money and having to wait more than 7 weeks for a machine, which then came without disks. Eventually, Commodore managed to sort that one out, but not without many calls and a telegram to Sydney.

A few months back, Commodore sent a disk of IFF stuff, and some documentation updates. Developers had this stuff already, maybe it was just a mistake. They also asked us to return a form if we needed the documentation on the expansion connector and Zorro bus. Of course, I said. Send it. I'm still waiting. Then, in September, they sent me a subscription form for **Kickstart, the European Amiga Technical Journal**. I sent a cheque by return mail for an airmail subscription, and I **still** haven't got anything from them. This is about the level of service I've come to expect from Commodore Australia, who seem to be more of a hindrance than a help to developers. Few local developers have received V1.2 AmigaDOS unless they've rung Commodore to hassle for it. In fact, most developers I've talked to seem to get nothing at all from Commodore unless they ring them and beg for updates and documentation. Perhaps they get sent things so they'll stop bugging Commodore, who obviously have better things to do than help the people who are helping them sell machines.

Anyway, on to lighter topics. Since I have to "catalog" the public domain disks for publication in the newsletter, I decided to spend some extra time and produce some **Public Domain Catalogs** for the group. I have put them in folders so they can be updated as new disks arrive. The catalogs will be kept with the library, and can be examined at AUG meetings. Since there will be a complete catalog in the **1986 AUG Yearbook**, I can see no real need to offer public domain catalogs by themselves at the moment.

Speaking of the **YearBook**, I haven't yet had the time to check with the printer about the price. We intend to get about 200 copies printed, sometime in February. The yearbook will contain reprints of everything from our 1986 newsletters, along with an up-to-date public domain catalog. Everyone who has been hassling me for back issues lately is expected to buy a copy!

Also just in is an Amiga **Instruction Course Videotape**, covering WorkBench and Introduction to the CLI. The tape is from Clackamas Computers in the USA, and is in NTSC VHS format. We hope to get the tape converted to PAL soon, and show the tape at the February meeting.

Recently, I bought the DOS 2 DOS utility program from Central Coast Software in the USA. The program ended up costing $94 after currency conversion etc. DOS 2 DOS allows you to read and write PC/MS-DOS disks on the Amiga using an external 5 1/4 inch drive. I wasn't going to buy this program, my initial choice was for DosDisk from Metadigm, Inc in California. When I buy software from overseas, I try to deal with companies who have run adverts for quite a while; there is a better chance that they aren't going to rip me off. Metadigm have been running adverts since Amiga magazines were first published. So, I rang them. The girl who answered the phone said all the sales people were out, and she couldn't take my order. So, I rang back the next day. And the next. Five times, I rang back. Every time the same story. Phone calls to the USA cost $2 a minute, and it looked as if I'd be spending more than the cost of the program on phone calls! On the last call, the girl took my number, and said that she would get a sales person to call me back, even though I was calling from Australia. After a week of waiting, I decided to ring the DOS 2 DOS people, even though I'd never heard of them and they'd only run one advert. A week later, I had the program in my hands! Great service, guys! I'll probably review DOS 2 DOS next issue. Metadigm: I really wanted to buy your program. How come you didn't want to sell it to me?

Other User Groups

So far, we've managed to locate three other Australian Amiga groups, in Adelaide, Brisbane and Canberra. We've put them all on our mailing list so we can exchange newsletters. Here are their details:

Amiga Mag
PO Box 486
Glenside, 5065
South Australia

Brisbane Amiga User Group
PO Box 853
Toowong, 4066
Queensland

Canberra Amiga Users Society (CAUS)
68 Wollongong Street
Fyshwick, 2609
ACT
(Bulletin Board: (062) 59 1137)

I've often wondered what else we could have called our group. It didn't take me long to decide against MAUG (Melbourne Amiga Users Group) which sounds like morgue, and VAUG (Victorian Amiga Users Group) which could be mis-pronounced as vague. The word Society instead of Group would have confused us with AUSOM (Apple Users Society of Melbourne), and I couldn't think of anything else. With visions of world domination, I decided not to narrow us down to a particular geographic area. So, we're called AUG.

1987 Meeting Dates

We've just sent the form off to book our 1987 meeting dates. Unless someone else has already booked the venue, here are the dates:

February 8th	June 14th	October 11th
March 8th	July 12th	November 8th
April 12th	August 9th	December 13th
May 10th	September 13th	

Pencil these dates into your calendar or diary, but make sure you look on the front page of the newsletter each month for any last-minute changes.

Well, that's about all I've got to say. See you at the **Amiga Users Group Barbecue**, 12 noon on Saturday January 17th, Studley Park, near the boat sheds.

-- Peter Jetson

AUG Bulletin Board

All is quiet on the BBS front at the moment, while we consider our finances. To get even a minimal system going, we need to spend about $2000. For a workable and useful system, $3000 is closer. Unless we can get someone to sponsor our system, or we can arrange a "long-term-loan" of a PC clone with a hard disk, I think we've got a way to go yet.

Software-wise, I think that Fido would serve our purposes best, allowing us to participate in this world-wide network of bulletin boards. If other Amiga groups were to make the same choice, the interchange of information about the Amiga would be made much easier, and Fido would bring the groups closer together.

For those who don't know, Fido and Fido-Net allow you to send messages to other Fido bulletin boards, anywhere in the world. Fido-Net carries Fido-News around the world, and Echo-Mail allows conferences or message areas to be "echoed" to other participating systems. In other words, messages left on our system could be sent to other Amiga groups, and vice-versa.

If you have a spare (or little-used) PC or clone hiding in your cupboard that you could loan us, or you or your employer might be interested in sponsoring the AUG bulletin board, please let me know.

Public Domain Update

Six new disks from the Fred Fish collection arrived on my doorstep just prior to Christmas, as well as a disk from **The Amigan Journeyman and Apprentice** magazine. The Amigan disk contains the latest version of the **Blink** program, as well as cute contribution called **Lens**. If you can imagine looking at your Amiga's screen with a magnifying glass with selectable magnification, then you've imagined this program.

Fish Disk #41 contains an update to my favourite terminal program, **vt100**, an adventure program generator, an update to the C shell, and a program to change fonts that I didn't manage to get running. Disk #42 is (yet another) Emacs for the Amiga. On disk #43, there is a very useful on-screen clock, **SpriteClock**. This clock pops onto the screen every 30 secs or so, and reminds you of how late it is. Also on #43 is **PopColours** which allows you to change your screen colours anytime you want from (virtually) any program. An interesting concept is **STEmulator** which allows your Amiga to pretend to be an Atari ST! (Read the **README** file on the disk before getting **too** excited).

On Fish Disk #44, we have some cute Icons, and some of the Ray Trace pictures from disk #39, saved in IFF format so they display quickly. Also **ViewILBM** that displays pictures until closed, instead of just for a few seconds. Disk #45 has quite a few interesting pictures from various artists, and **WhereIs** that locates programs wherever they are on a disk. Lastly, disk #46 contains two completely useless programs, **ValSpeak** and **Jive** which translate input files into two American "dialects". On the up side, it also contains a **public domain assembler** and a **Gadget Editor**.

Unfortunately, the **Amicus Disks** have still not arrived. Perhaps the bank draft or the disks have gone astray, or they were sent surface mail. I'll ring **Amazing Computing** in the next few days or so to try to work things out.

All the disks mentioned below and in previous newsletters are available **NOW** from the Amiga Users Group. See the order coupon on the inside back page for more details.

Fish Disk #41

AmigaVenture	- A program which allows you to write your own Infocom-style adventure programs in AmigaBasic. It is a full-featured adventure parser, including direct and indirect objects, multiple object processing, adjectives, automatic ambiguity resolution, and subordinate clauses. The parser includes support for one, two, or three-word verbs, and a full set of object-manipulation primitives.
Csh	- Version 2.03 of Matt's Csh-like shell. Executable only.
Dbug	- Macro based C debugging package. Machine independent. Provides function trace, selective printing of internal state information, and more. First released on disk #2. This version includes some bug fixes and enhancements.
DualPlayField	- An example of using a dual-playfield screen, using a method contrary to documentation in the Intuition Manual.
GetFile	- A very nice file name requester. Unlike the earlier version on disk #35, this version includes source code.
LatticeXref	- A cross reference listing of all symbols defined in the Lattice 3.10 header files. Sorted alphabetically by symbol string, includes file name and line number of all references and/or definitions.
Lines	- A line drawing demo program, reminiscent of the "sparks" program on disk #9.
SetFont	- A program to change the font used in a workbench screen. Includes several sample fonts of various sizes.
Vt100	- Version 2.3 of the ever popular vt100 terminal program. Includes xmodem and kermit file transfer protocols.

Fish Disk #42

To quote the "Read Me First" file: This diskette contains the Amiga version of MicroGNUEmacs (MG), a small but powerful text editor that runs on many other computer systems besides the Amiga. One of MG's major goals is to be compatible with its cousin GNU Emacs, so certain features you may have seen in other versions of MicroEmacs may work differently here, or not exist. Hopefully, you'll find the added features MG provides to be worth the trouble it takes to make the switch.

Fish Disk #43

BasicBoing	- An AmigaBasic program which shows animation by page flipping. Precalculates all views of a rotating 3 dimensional cube and then cycles through them rapidly for animation.
Bbm	- Demo copy of B.E.S.T. Software's Business Management System. It is a full implementation with file sizes reduced for demo purposes.
BbsList	- A list of Bulletin Board Systems which support the Amiga. The list was compiled from a list on Delphi, Compuserve, bathroom walls, etc.
Cc	- C compiler frontends for Manx and Lattice C, developed independently by Jay Ts. These automatically filter off the annoying banner messages from various passes of the compilers.
Copper	- A copper list disassembler. Dumps the contents of a hardware copper instruction list.
InstIFF	- A program which converts sampled sound files from the Instruments dealer demo disks to IFF sampled sound files in a FORM 8SVX.
PopColours	- Lets you change the Red/Green/Blue components of any color register, on any screen currently in the system. Uses a movable window with slider gadgets. Very well done. Version 1.0, November 1986.
SpriteClock	- A very simple clock that uses a sprite as it's display medium, thus allowing it to be displayed on top of all other screens. Includes source in assembly language.
STEmulator	- Turns your Amiga into an Atari ST (sort of). Be sure to read the README file for the true story...
WBrun	- A program designed to allow any program to be invoked from CLI yet behave as if it were invoked from Workbench. Workbench need not be loaded, thus saving the memory that Workbench would normally use.
Wild	- Two versions of Unix shell style wildcard matching routines.

Fish Disk #44

Icons	- Some miscellaneous icons for your viewing pleasure.
NewIFF	- Some new iff material dealing with sampled voice and music iff files.
RayTracePics	- Ray tracing pictures, some of which appeared on disk number 39, but now in IFF HAM format for MUCH faster loading and compatibility with existing IFF tools.
ViewILBM	- Reads an ILBM file and displays as a screen/window until closed. Handles normal and HAM ILBM's.

Fish Disk #45

Clue	- Clue board game. Nice.
Make	- Another version of make that seems to be more complete than many other PD makes.
Pictures	- Miscellaneous pictures selected from dozens of pictures that have come my way since the last full art disk.
Update	- Used to update an older working disk with files from a newly released disk. Files on the older disk that are out of date will be upgraded with files from the new disk.
WhereIs	- Program which searches a disk for the first or multiple occurances of a file with a given name.

Fish Disk #46

Asm	- A shareware macro assembler, submitted by the author. Asm is a 68010 macro assembler that is compatible with the assembler described in the AmigaDOS manual.
CheckModem	- A program which provides for executing other programs from your startup file, if and only if there is actually a modem connected to the serial port.
Egad	- A gadget editor from the Programmers Network. Very nicely done and very useful.
Jive	- A filter program which transforms its standard input to "jive" on its standard output.
My.lib	- A binary only copy of Matt's alternate runtime library.
ProffMacros	- Subset implementations of the Berkeley "ms" and System V "mm" macro packages, for the proff program.
ValSpeak	- A filter program which transforms its standard input to "valspeak" on its standard output.

Amigan Disk #4

Make	- A utility to help organize your project and keep everything up to date. This version is only 14k, but it incorporates all the most important features of the UNIX (tm) version, including macros and default rules.
Blink	- A linker designed to replace ALINK. BLINK is public domain, is faster than ALINK, produces smaller code than ALINK, produces a more easily readable object file map than ALINK and takes up about the same amount of space on your disk (31k vs. 29k). BLINK also has support for some new features that will be out in Lattice C version 3.04 which will allow the compiler to produce much smaller code.
IconTools	- This is a set of four utility programs taken from a PD disk distributed by the New York Amiga user group AMuse. The first utility is ZAPICON, which lets you take a Deluxe Paint brush or an Aegis Images window and turn it into an icon. ICONEXEC ties an icon to a specific command. Any command executable from AMIGADOS can now be executed from the workbench! SETALTERNATE allows you to give the icon a new picture when it is selected by a mouse click. This gives rise to some really great effects. SETWINDOW allows you to set up the output window your CLI program will use when executed from workbench.
Arc	- ARC is a utility program widespread on bulletin boards all over. It takes a number of files and compresses them as much as it can, then lumps them all together in one file for easier handling. Many BBS's have files marked <name>.arc; these are ARC libraries, and should be unpacked with ARC.
FixObj	- Fixes a common problem when downloading software from a bulletin board. XMODEM, the most widely used terminal protocol, must have chunks of data that are exactly 128 bytes long. If the last chunk of your program is not exactly 128 bytes, XMODEM pads the data out to 128. This technique works fine on CP/M systems and on the IBM PC, but unfortunately NOT on the Amiga. If you try to execute one of these padded programs, AmigaDOS will respond 'not an object file'! FIXOBJ fixes these padded programs so they will run on the Amiga.
PopCLI	- A super little utility, POPCLI has two functions. First, if you haven't typed anything or used the mouse for a given period of time, POPCLI will turn your screen black to save the phosphors. It turns on again if you touch a key or the mouse. Second, any time you want a CLI, in any program, all you have to do is hit left-Amiga Escape and voila!
Tsize	- This gives you a listing of the number of bytes used by each directory on a disk. Very handy for seeing where the waste comes from!
Qcopy	- A graphical DISKCOPY. It shows what sectors it has copied and what sectors it had problems with. QCOPY can recover from errors that DISKCOPY gives up on--and you can copy a series of disks without re-issuing the QCOPY command. Be warned: it's okay with 512K or less; it crashes with expanded memory.
Monopoly	- A great old game; in ABASIC; it's stuffed with graphics and can handle up to four players. Enjoy.
Epfix	- A patch program which fixed the graphics problems in the standard AmigaDOS V1.1 Epson printer drivers.
Lens	- Run this program! It creates a "magnifying glass" view of the current screen.

Amiga Windcheaters

First the bad news: we've sold out of our first lot of Amiga Windcheaters. The good news? We're about to order another lot!

Quality, fleecy lined, white windcheaters with a full color Amiga logo on the left breast are just $25. That's about the same price you'd pay for a plain windcheater.

To order, contact our purchasing officer, "Drac", at the next meeting, or ring him on (03) 792 1138 (ah).

AMIGA Dealers:

Why not advertise in

AMIGA WORKBENCH

An advert this size costs only

$20

(from camera—ready artwork)

Other sizes are available:

Half Page:	$40
Full Page:	$70
Double page:	$120

AMIGA Users Group
PO Box 109, North Balwyn, 3104

Editorial

I'm a bit disappointed at the lack of contributions to this month's newsletter. Perhaps everyone went on holidays. Whatever the reason, **don't let it happen next month!**

A few people have mentioned to me that a lot of the articles we've been publishing go right over their heads, and that we are getting too technical. As editor, I'm constantly aware of this problem, but I can only publish what I'm given, or write myself.

Since I'm into the Amiga in a technical way, that's what I write about. I can't write about AmigaBASIC, since I rarely use it. If you use it, then write an article about it. I can't write about Workbench, since I use the CLI almost exclusively. Why don't **you** write an article about the Workbench?

I can write about C, or detail some of my head-scratching when I delve into the Rom Kernal or the Amiga's hardware. In fact, I've been going to write an article about using the Amiga's printer port for general purpose IO, and another describing the design, construction and software for a battery-backed clock I've built for the Amiga. I haven't written the articles because people are complaining about how technical the newsletter is getting.

So, I put my articles on the back-burner and write editorials that will (hopefully) get people to write about things that everyone might like to read. It's not hard to write an article. I'm sure that everyone in the club knows something that might answer someone else's question. If you've written a program for the Amiga, even something simple, then write an article to go with it. You'll be helping someone, and that's what the club is all about.

If you've recently bought a book about the Amiga, then review it for the newsletter so everyone else can benefit. The same goes for software. Write a review on your lastest purchase.

I've often found out about something, only to find that everyone else already knew it. Don't assume that because I write the newsletter, I know everything Amigan. I live 20 miles out of town, and only make it to an Amiga dealer every few weeks. Many of you hear about Amiga news or gossip long before I do. Write it down and I'll publish it! If you work for a store that sells Amiga software or hardware, drop us a note to tell us what's new. Everyone would be interested in that.

-- Peter Jetson

SOFTWARE ORDER FORM

Disk numbers :									

Disks supplied by Amiga User Group @ $10	$
Disks supplied by member @ $2	$
Club Use Only Receipt #: Mailed on: / /	Total $
Mail to: Amiga Users' Group, P.O. Box 109, North Balwyn, 3104.	
Member's Name:	
Address:	

Application for membership of The Amiga Users Group

Membership is $20 per year. Make cheques payable to The Amiga Users Group, and send to:

Amiga Users Group, PO Box 109, North Balwyn, 3104

Details this side are optional

Surname: ____________________

First name: ____________________ (no initials)

Address: ____________________

____________________ Postcode: ________

Where did you hear about AUG: ____________________

How can AUG help you: ____________________

____

____

Signed: ____________________ Date: ____________

In the event of my admission as a member, I agree to abide by the rules of the Association for the time being in force.

Phone number: ____________________ STD code: ________

Year of birth: ____________________

Occupation: ____________________

Interests: ____________________

____

____

Dealer's Name: ____________________

Dealer's Address: ____________________

____________________ Postcode: ________

Are you happy with your dealer: ____________________

AMIGA Workbench
Registered by Australia Post
Publication No: VBG7930
If Undeliverable, return to:
PO Box 109, North Balwyn, 3104
Victoria, Australia

Postage Paid
Caulfield
East 3145

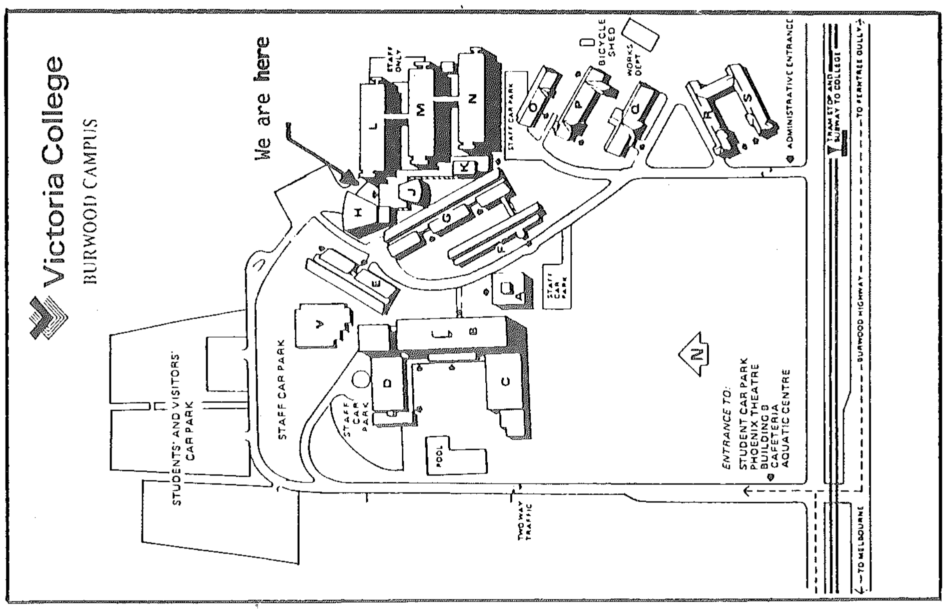
+ Member # 10297   Expires end of 11-87

Campbell Gray
RMIT Comm Eng Dept
Box 2476V
Melbourne   3001

AMIGA™ Users Group

P.O. Box 109, North Balwyn, Victoria, Australia, 3104

January 1987 Amiga Workbench



Where is Victoria College Burwood Campus?

New members and visitors sometimes have trouble locating our meeting place the first time. Victoria College is on the North side of Burwood Highway, Burwood, just East of Elgar Road. Coming from the City, turn left at the first set of traffic lights after Elgar Road. Follow the road around past the football oval, over three or four traffic bumps to the car parking areas near the netball courts. Further up the road, to the left, you'll find Lecture Theatre 2. That's us!

If you have a Melways, try Map 61 B5.