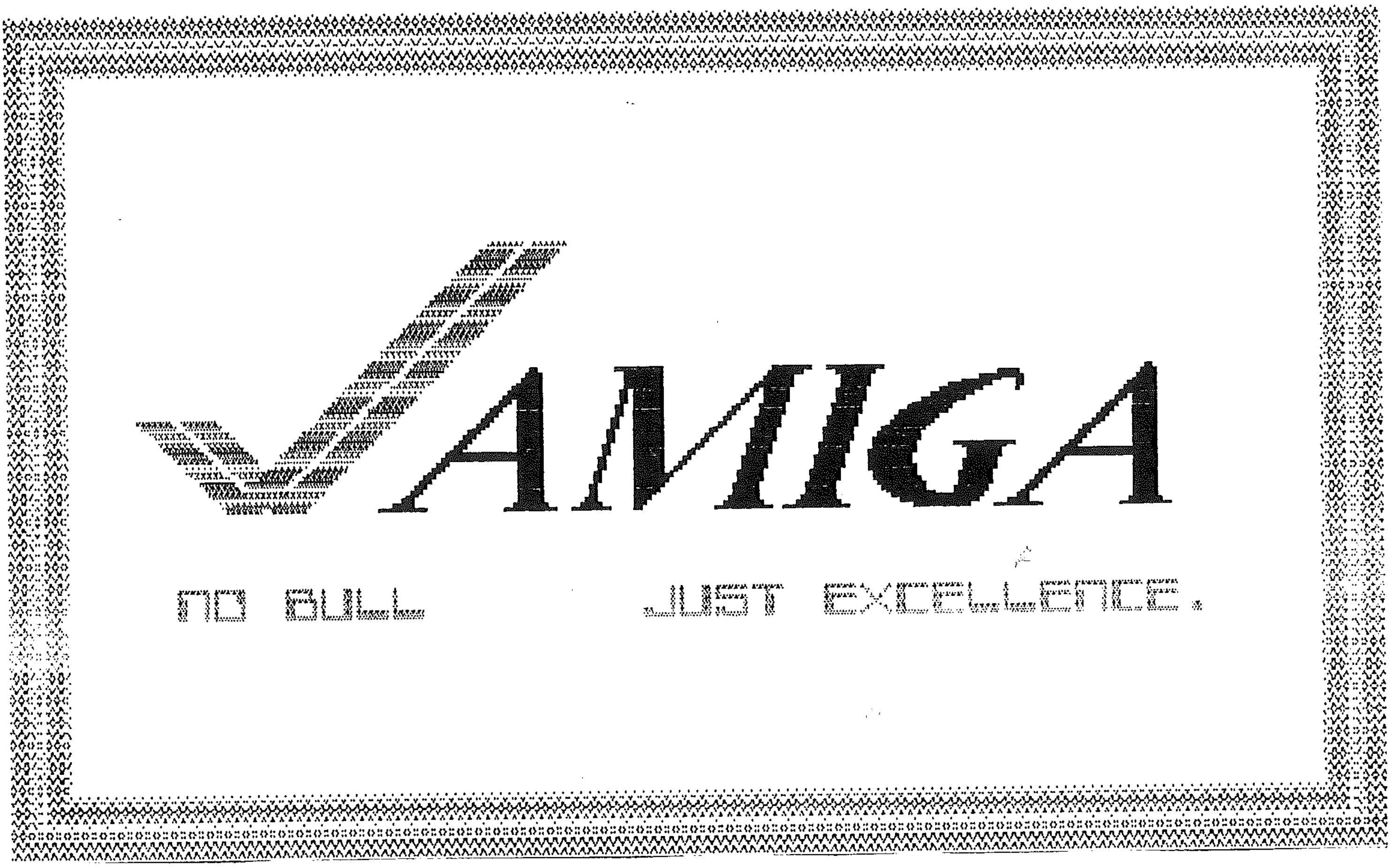
AMIGA

WORKBENCH

$2 RRP

Registered by Australia Post – Publication No. VBG7930

Number 20 | Circulation: 1000 | January 1988



Next Meeting

No formal January meeting. Instead, come to our Barbecue on Saturday, January 16th from 12 noon onwards.

The January meeting will be held at Bundoora Park,
Plenty Road, Bundoora – Melways map 19, reference F4

Amiga Users Group Inc, PO Box 48, Boronia, 3155, Victoria, Australia

Dark Castle Review

by Nigel Harwood

The Dark castle box contains two discs and a card containing a brief description of the commands available. The first thing which struck me was the title on the card, "Dark Castle for the Amiga". I asked the shop assistant about this and he told me that Dark Castle had previously been brought out on the Apple Macintosh.

I am of the opinion that a game which has been released for other machines in the past, and has now been ported to the Amige, is usually a game with some merit. The only thing which I have found that I must watch is that the game has REALLY been ported. That is, the Amiga's capabilities have been used where ever possible, and in this regard Dark Castle has done quite well.

The game revolves around you controlling the hero, and you attempt to pass through various areas of a castle, defeating multiple nasties (rats, bats, mutants etc etc) by throwing rocks at them. As you go you can pick up things like more rocks, anti rat bite potions, keys etc. Your ultimate goal is to overthrow the black knight, but first there are fourteen areas to pass through.

Having played Dark Castle for a while, I have decided that I like it, for the following reasons.

1. Unlike games like Barbarian, Dark Castle offers a lot more flexibility in play. For instance, there are many ways to pass through an given area. In fact you can elect at the start of the game to enter the areas from four different points.

2. Because this game has had time to mature, due to its previous versions, all the extras (which I think are in fact necessities) have been included, like saving high scores to disc, demo mode, a set of very good info/help screens, three levels of play, clear high scores, etc.

3. Good sound effects, which I doubt would be as good on any other machine. Animation and graphic detail are also fairly good.

4. Good interest level, with every screen sufficiently different to present a new challenge. Also special items are stowed in various areas, like the magic shield, fireballs and mace, with which you tackle extra nasty nasties like whipcracking henchmen, Merlin the magician, dragons etc. Overall a lot of thought and imagination has gone into it, which all results in more pleasure for the player.

My conclusion is that Dark Castle is a good quality game, not perfect but certainly one of the best of the latest releases.

==

Our Bulletin Board - An Open Letter

from George Wahr

Tried to get onto the Amiga Users Group Bulletin Board lately and not succeeded? I have. I've even met people who have been trying for weeks without luck. I believe that next to meetings and this newsletter, our BBS is the most powerful tool the club has to promote itself, while also provide meaningful benefits to club members. And yet it is unable to function properly because it is so popular that it is killing itself as people give up trying to get on.

I believe part of the blame lies with Drac, who has done such an excellent job running the board, making himself available to users at all hours of the day and night, as well as taking on the added responsibility of running our CommSIG. His dedication has no doubt played a large part in the success of the board and therefore of the Amiga Users Group itself.

With such a large number of people voting with their modems and demonstrating the popularity of the board, I think the club should consider getting another BBS. I know that some people are going to jump up and down and say "We can't afford it", but the reality is; can we afford not to. If the club is going to grow and expand, we must expand the services offered to members. I think that if we outlay money in order to expand better services to members, we will recoup this outlay in increased membership.

Another board could be organised very cheaply by getting another PC clone/modem combination and networking it to our existing BBS, thus having two phone lines at Drac's place. Another option would be a whole new BBS at a separate location. This would be slightly dearer as an additional hard disc would be required. New files and messages could be cross sent between the Boards during mail hour so that for all intents and purposes, the boards would appear as one.

One way of offsetting the cost could be to take on sponsorship, in much the same way that we charge traders at club meetings to exhibit and sell their wares. (I know this is a contentious issue and I only put it here as an option to be considered, as I also see benefits in the club being independent. I hope it does not distract people from the point I am making, that we need to get a new BBS.) This could be done on the board where traders could advertise there products and special offers in a special section. It would have absolutely no effect on anybody except those who wanted to access the section to find out about these offers or products. Money from this could be used to offset the cost of commissioning and running the new board.

In summing up, the BBS is something that I get a great deal of enjoyment from and I feel that it is terrific that it is now so busy that people find it hard to get on. I would like to see more people benefit from the BBS, and I believe that now is the time for the club to act and introduce a second board. I would be happy to hear from any one about my proposal for a second board. Ph 376-6180 or leave a message on the BBS.

==

AmigaBASIC Blues

by Evan Stamatopoulos

First of all, I would like to praise those responsible for the production of this newsletter. It contains many interesting and informative articles which make it mandatory reading for every Amiga user. [Thanks - ed] It was while I was going through one of the previous newsletters that I came across a plea by the editor for more contributions to the newsletter of a less technical nature. Since then I have noted more articles like hardware/software reviews etc. Well, about half a year after this plea, I have decided to finally overcome my laziness and write an article about AmigaBASIC.

About five months ago, I was writing a BASIC program which could be used to define 3D solids and to rotate them about the x, y or z axes by an angle specified by the user. As there were a lot of calculations involved, I tried to cut down on calculation time by defining variables as either integers or long integers. No matter how hard I tried to debug my program, it just refused to work. I checked my equations over and over again but found absolutely nothing wrong with them. I then sat back, took a long deep breath, and started to think about the problem logically. "If my program is correct", I thought to myself, "then it stands to reason that Microsoft's software (i.e. AmigaBASIC) must be at fault". So I set out to test this highly unlikely assertion.

I started playing around with integer and long integer numbers in the output window. After a little while I typed in ?-1*100000& and lo and behold the answer was 100000 instead of -100000. It seems that if a long integer is over 64k (2 bytes long) then multiplying it with a negative number gives you a positive number.

I had to change my program so that some of the equations were calculated using floating point arithmetic. This fixed up the 'bug' and the program worked fine. Unfortunately AmigaBASIC still didn't. So I waited with bated breath for the version 1.2 software to be released. To my surprise, this also had the same error in its long integer arithmetic. So I am now writing to AUG in the hope that somebody out there can debug AmigaBASIC and to warn others to be wary when using long integers.

Apart from this problem, I have found AmigaBASIC to be the best BASIC interpreter out of all those I have used (e.g. IBM, MacIntosh, 8-bit machines, VAX). AmigaBASIC has features that the others just don't have, like labels instead of line numbers, subprograms, IF..THEN..ELSE..ENDIF constructs which execute over multiple lines, WHILE..WEND, easy access to O/S routines and libraries, etc. It's even faster than the BASICs on most other micros.

Having said that, I would like to point out that I use C more than I use BASIC because it's a more powerful language and it runs much faster (even if you compile your BASIC programs with something like AC/BASIC). The only advantage that BASIC has over C is that it is an interpreter and it therefore takes less time to develop a BASIC program. That's why I use BASIC for small non-intensive programs (e.g. drawing graphs, etc). What I would ideally like to see is a C interpreter (which I believe are available for IBMs), so I can develop a program quickly then compile it when I have finished. Then I wouldn't need to use BASIC at all.

===

ATTENTION

Residents of the North-Western Suburbs of Melbourne

Are you new to the Amiga?

Do you have problems running your Software and/or Hardware?

Would you like to meet other Amiga Users in your area?

Do you want to find out what makes the Amiga tick?

I am writing this to see if there is a demand for a informal monthly meeting of interested Amiga Users in this area. This will NOT be a "Splinter" users group, but a back-up group for AUG for those members, and non-members, who find they cannot attend the AUG meetings for one reason or another. If there is a demand, I will arrange a meeting at a future date for those interested.

Please drop me a line or ring me, and I will send out more details when they are available.

Neil Beatty
PO Box 204,
Essendon, Victoria, 3040.

You can phone me on (03) 370-9976 between 7 and 9pm, weekdays and Sundays.

===

Test Drive Review
by Allison Greer

People born on Christmas Day pose a problem when it comes to buying that special present for the 25th. I have that problem with a friend. I had a bigger problem keeping his present under the tree!!

I saw a demonstration of Test Drive and knew I would enjoy using this on the Amiga, so I brought it for him, and then pestered him for hours to open up the parcel. Finally he gave in to my demands, and I played with it for hours, scoring in excess of 21,000 on the first night. He scored 8927.

One of the most important features of this game, in my case, is the price - only $49.95, Very cheap and easily one of the best games I've seen. The graphics are above average, although lacking in variety (more later). Audio is excellent, from the introduction with its catchy tunes through to the sounds of the engines. Game play is via a joystick, and it really needs no instruction as there are basically two moves, gear change and accelerator. (Who uses brakes?)

You can "test drive" five different cars - Chevy, Porsche, Ferrari, Lotus and a Lamborghini, all of which are manual gear change, with 4 or 5 gears plus reverse which is not used.

The surprising thing about this simulator is how different each car is to drive. I won't say how I like the (insert a car name here) because of it's grip on the road, or how I hate the (insert another car name here) because it slides across the road like a tram without rails. I will let you do that. I will say to those who do not have a driver's licence, start on the Chevy first to get the feel of the car, road and gear changing.

Now for the gripes. It's Right Hand Drive!!!! Boy, I bet that truck driver hates me because of the accidental meetings we have on a particular curve. After doing driving lessons at school and recently obtaining my "L" plates, it is weird to have someone yell out "Allison, you aren't on the AMIGA now, get on the LEFT side of the road." Oh well, back to the back seat.

The graphics tend do get a bit boring after a few hours of driving, and I hate the frustration that builds up because I can't obtain that goal of topping 183 mph because I'm always on the curvy mountain road.

Yes, your Amiga can be pulled over and booked for speeding. You have to either speed away from them or "take it like a man" when the boys in those "bubble cars" wish to speak to you. Make sure you don't run up their rear, or you'll end the game. Also, don't be a "goody goody" and stick to the speed limit, you will pay the penalty otherwise. (If this is too fast try "Flight Simulator II".) You have some of the fastest cars at your fingertips, use them that way.

To quickly sum-up, this is a game ALL members of the family, even Grandma, will enjoy playing. Fantastic sound, good graphics and Great Fun. Go out and buy it!

Now, I wonder when my driving instructor will be over his case of nerves and allow me to resume my lessons. I was only going 15 kph over the speed limit, and I missed the bus by at least six centimetres.

===

Handy Ascii File Peeker
by Mark Kelly

You are browsing through a Fish disk. You see an interesting filename. Is it a binary file or an EXECUTE batch file? You don't know. (I, for one, would be grateful if programmers used an '.x' suffix or something on batch files so others can tell what it is!)

You TYPE it, hoping it's all ASCII, and get hieroglyphics. You grumble as you TYPE it again with the OPT H option and scan the narrow column of ASCII characters beside the wide and unuseful hex dump column. Wouldn't it be nice to use a single command to inspect any file?

Enter DUMP. DUMP will display any ASCII characters (and CR's and tabs) in the file and replace unprintable characters with full stops. DUMP is useful whenever you want to see the ASCII content of a file (such as when you are fed up with an adventure game and want to scan it for hidden vocabulary and clues about what you can do.) DUMP runs pretty quickly so use the space and backspace keys to pause and restart the ASCII dump.

Unfortunately my C programming hasn't reached the stage where I can trap control-C's to break execution so you can't stop DUMP until it hits the end of its input file. Any clever bunnies who can patch in break-trapping (and send it to WORKBENCH) will earn my undying gratitude and a Freddo Frog.

The source code is alongside this column. Happy hacking!

```
/* DUMP -- dump file in ASCII by Mark Kelly.
   Usage ...  DUMP filename */

#include "lattice/stdio.h"
FILE *infile;

main(argc,filename)
int argc;
char *filename[];
{
   register char c;
   if ((infile = fopen(filename[1],"r")) == 0 ) {
   printf("Can't open %s\n",filename[1]); exit(1); }

   for (;;) {
   c = getc(infile); /* get a byte */
   if (feof(infile))
      { putchar('\n'); exit(0); }
 /* end of file?  */
   else if (c == '\n' || c == '\t') putchar(c);
 /* print CR's & tabs */
   else putchar(c < ' ' ?  '.': c);
 /* neat, huh?  */
   }
}
```

===

===

Forming a Special Interest Group
by John Elston

I am now going to admit to something that took a great deal of courage to own up to. I have finally come out of the closet. I am no longer going to live a life of deceit and subterfuge, I am going to bare my soul to the world and stand tall and admit that I use BASIC and enjoy it!! Gasp, shock, horror, I hear you say. Someone actually uses Amiga BASIC, and worse still, he admits to enjoying it! What brings me out into the open after years of hiding, afraid to let anyone examine my disks in case they discovered my awful secret you ask?

It all began at the November AUG meeting when I foolishly mentioned to Bob Scarfe (the AUG coordinator) that someone should start a BASIC SIG. Bob, being a practical sort of person, agreed and announced at the main meeting that a BaSIG would be being formed and anyone interested should see me.

Due to my spur of the moment decision to start the SIG, there wasn't a room allocated, so an area between two of the rooms at VICTRACC was allocated for the BaSIG. There were several chairs and a table, more than adequate for the anticipated person who might arrive, I thought.

The appointed time for the meeting arrived and lo and behold seven people turned up (approximately seven more than I anticipated). Now the problem was here were seven people expecting someone who was supposed to know something about special interest groups and how they are run and all they had was me.

Never having spoken to such a vast horde of people before, I girded my loins (metaphorically speaking) and said something to the assembled throng. "Welcome to the BASIC SIG". So far so good, I thought, but what now? Inspiration hit me, BOING! (Inspiration always strikes me like that.) Appeal to them for help! So, clearing my throat for dramatic effect I said "I have never done this sort of thing before, please bear with me, what do you think a BASIC SIG should do?"

Fearing a dreadful pregnant silence after these words, I waited with bated breath. However, much to my chagrin, I could not get another word in edgeways. Everyone was tremendously enthusiastic, full of ideas and prepared to voice them. This was incredible. Here were calm rational people who, like me, not only programmed the Amiga in BASIC, but did so unashamedly and were prepared to share their thoughts, ambitions and problems with each other! Perhaps there is hope for me yet. The meeting ended with everyone agreeing to bring in a sample of their handiwork to show to everyone else at the next meeting.

Bouyed up by the success of the first meeting, I approached the December meeting with trepidation and anguish. What if I made a complete idiot of myself and no one attended this meeting? (I worry a lot.) In case no-one attended with their programmes, I copied some public domain BASIC programmes onto a disk for demos for when I ran out of words.

Again I was pleasantly surprised at the response, now there were something like ten maybe twenty people. OH MY GOD! Do I have to speak to this enormous crowd? These people they are all going to look at me, what am I going to do? Where's the nearest bar? Steady on, I thought, you have already been through this before. Seven or seventy, what does it matter? You survived the last meeting, you can do it again can't you?

Doing my stuff with my loins again, I said "Welcome to the BASIC SIG. Does anyone have any questions?". To my eternal relief, people responded with interest and enthusiasm again and we were off and running. We saw some interesting demos and everyone seemed to get something from the meeting. Unfortunately we were between two rooms again, but I hope to correct that next meeting by arranging for room to be allocated for us.

The December meeting produced several ideas for the BaSIG, such as BASIC tutorial for beginners, a disk of subroutines and a disk of public domain programmes.

I would like to thank everyone who attended the BASIC SIG meetings, I hope that they found something of value from each one and will continue to attend future meetings. I hope the SIG will become a little more organized and structured as time goes by, and judging from the enthusiasm shown by everyone this will be the case.

If you have any (polite) suggestions or questons about BaSIG I may be contacted on (03) 375 4142.

===

Flight Simulator II Null-Modem Connection

by Richard Black

Parts List:

2 x Amiga computers
2 x SubLogic Flight Simulator II programs
2 x would-be pilots
Some spare time
Some 240 VAC power outlets
2 x DB-25P RS-232 connectors
2 x DB-25 Backshells
Some 'hook-up' wire
Some 3 conductor cable

Given that the most powerful thing God ever made is generally recognised as the empty Coke bottle, the second most powerful must be two Amiga's playing FlightSimII together. To experience this truely scrumptious event for one's self, simply purchase the parts listed above and then muddle your way through the assembly instructions in paragraph two.

After obtaining all the parts listed above in the parts list, begin your adventure by jumpering pins 4 and 5 together on each plug. Next jumper pins 6, 8 and 20 together on each plug. Next, out comes the three conductor cable. Connect one conductor between Plug 1 pin 2 and Plug 2 pin 3. Connect another conductor between Plug 1 pin 3 and Plug 2 pin 2. And finally connect the third and final conductor between Plug 1 pin 7 and Plug 2 pin 7. Now fit the backshells, boot FlightSimII on both Amiga's and select multi-player mode.

Reference: Elektor Magazine November 1985

===

BY COMMODORE® AMIGA TM

Advanced Graphics SIG

by Temporary coordinator Enno Davids

At the last meeting of our SIG we built on some of the questions asked in our first meeting, in the previous month. Short talks were presented by several members on two main topics. The first of these topics was a general one in which several members were asked to share with the rest of the group the steps necessary to open a window on the Workbench Screen and to plot a point in this window. The languages chosen were 68000 Assembler, C, and Modula II. These languages were mainly chosen on the basis of suitable "volunteers" being available.

The second topic was a background into the mathematics behind the transformation of points in a 3-dimensional coordinate space. As these primitives are the basis of operations on 3-dimensional objects, this is a valuable insight into the techniques needed for the manipulation and display of such objects. Due to the substantial amount of information we tried to present this last talk had to be prematurely halted to allow the Desktop Publishing SIG (the group scheduled for the next use of the room) to hold their meeting. As a result, the balance of this talk will be presented at the next meeting, with, hopefully a report in the next newsletter.

For the balance of this article I will present the C and assembly versions of the point plotting programs that were presented. In doing this I do not intend to explain how C or Assembly language work. These are better left to other avenues than this article. Firstly, some general comments on how these programs are constructed.

The actions these programs need to take to allow them display a point on the screen are of course the same. It is merely the language that is different. As you will see, and as one would expect, the assembly source is considerably longer than the C source. I will make some parting comments about the development times and so on later.

The point plotting programs both follow roughly the sequence of events shown below. Both programs are organised as basically linear programs that have no control statements (if's, loop's, etc.) other than those necessary to release the resources that they acquire along the way. So, if brief here is the sequence of events that both programs follow.

Acquire resources from AmigaDOS
Initialise NewWindow data structure
Create window
Select "pen" color & plot point
Close window
Release resources back to AmigaDOS
Exit gracefully

Each step alone is fairly straightforward, although those who wish a thorough understanding will need to invest in a set of Amiga manuals. It is unfortunate that most of these refer mainly to Version 1.1 of the system software and that information is strewn about these manuals in a fairly haphazard manner. These in themselves are not major hindrances but merely serve as yet another stumbling block when one tries this for the first time. More significantly, the manuals are slanted very considerably toward the C programmer. As a result those wishing to fully understand the assembler example will have harder work. As a guide, the Assembler example program was developed directly from the C program and thus the C can be used as a guide to Assembler.

```
/*
 *-----------------------------------------------
 * Sample program to set a point in a window opened
 * on the workbench screen.
 *
 * :ts=8
 */

#include <stdio.h>
/* Get some system data structures */
#include <intuition/intuition.h>

struct IntuitionBase*IntuitionBase;
struct GfxBase*GfxBase;
struct DosBase*DosBase;
struct Window*win;

main() {
struct NewWindownw;
struct RastPort*rp;

/*
 * Open all the libraries we will need
 */
IntuitionBase = (struct IntuitionBase *)
OpenLibrary("intuition.library", 0);
if (IntuitionBase == NULL) {
puts("Couldn't open Intuition Library");
exit(1);
}

GfxBase = (struct GfxBase *)
   OpenLibrary("graphics.library", 0);
if (GfxBase == NULL) {
puts("Couldn't open Graphics Library");
CloseLibrary(IntuitionBase);
exit(1);
}

DosBase = (struct DosBase *)
   OpenLibrary("dos.library", 0);
if (DosBase == NULL) {
puts("Couldn't open Dos Library");
CloseLibrary(GfxBase);
CloseLibrary(IntuitionBase);
exit(1);
}
/*
 * Now create the window we are interested in using
 */

nw.LeftEdge = 20;
nw.TopEdge = 20;
nw.Width = 300;
nw.Height = 100;
nw.DetailPen = 0;
nw.BlockPen = 1;
nw.Title = (UBYTE *)"Set a point in here?";
nw.Flags = SMART_REFRESH | ACTIVATE | GIMMEZEROZERO ;
nw.IDCMPFlags = NULL;
nw.Type = WBENCHSCREEN;
nw.FirstGadget = NULL;
nw.CheckMark = NULL;
nw.Screen = NULL;
nw.BitMap = NULL;
```

```
nw.MinWidth = 0;
nw.MinHeight = 0;
nw.MaxWidth = 0;
nw.MaxHeight = 0;

win = (struct Window *)OpenWindow(&nw);

if (win == NULL) {
puts("Couldn't open Window");
CloseLibrary(GfxBase);
CloseLibrary(IntuitionBase);
exit(1);
}

/*
 * We now find ourselves a pointer to the rastport we
 * wish to use.  We need one of these for the graphics
 * library calls we wish to make.
 */

 rp = win->RPort;

/*
 * We are now ready to plot a point on the screen
 */

SetAPen(rp, 1);/* Select the color to draw in.  */
WritePixel(rp, 10, 10);/* Draw the point.*/

Delay(50 * 5);/* A delay for us to see the point */

/*
 * Now we close down and free our resources in an
 * orderly manner.
 */

CloseWindow(win);
CloseLibrary(DosBase);
CloseLibrary(GfxBase);
CloseLibrary(IntuitionBase);
}
```

Now we come to the Assembler version of the above program. It is as previously stated a direct translation of the C program. The only difference is that the NewWindow structure is not allocated and initialised at run time as in the C code but is a static data structure in the assembler source file.

```
;
;---------------------------------------------------
; Sample program to set a point in a window opened on
; the workbench screen.  This is an assembler version
; derived from the C version.
;
; :ts=8
;

nolist
          include  'intuition/intuition.i'
; Get some system data structures
list

SysBase   equ  $4

;---------------------------------------------------
jsrlib    macro
          xref      _LVO\1
          jsr       _LVO\1(a6)
          endm
;---------------------------------------------------

          move.l    a0,d2
          move.l    d0,d3
          moveq     #20,d7          ; set up for a
                                    ; FAIL return just
                                    ; in case
          move.l    SysBase,a6      ; get ptr to
                                    ; ExecBase (for
                                    ; OpenLibrary, etc)
;
; Open all the libraries we will need
;
; Intuition
;
          moveq     #0,d0           ; we don't care
                                    ; which version
                                    ; we get (ie.>=0)
          lea       intuname(pc),a1; point to name of
                                    ; library we want
          jsrlib    OpenLibrary
          tst.l     d0              ; see if we got a
                                    ; good open?
          beq       AbortExit       ; no, so we fail
          move.l    d0,IntuiBase    ; save Intuition
                                    ; ptr for later
;
; Graphics
;
          moveq     #0,d0           ; we now do the
                                    ; same for the
                                    ; Graphics library
          lea       gfxname(pc),a1
          jsrlib    OpenLibrary
          tst.l     d0
          beq       IntuClos
          move.l    d0,GfxBase      ; save Graphics ptr
                                    ; for later
;
; Dos
;
          moveq     #0,d0           ; we now do the
                                    ; same for the
                                    ; Dos library
          lea       dosname(pc),a1
          jsrlib    OpenLibrary
          tst.l     d0
          beq       GfxClos
          move.l    d0,DosBase      ; save Dos ptr
                                    ; for later
;
; Now we create the window we are interested in using
;
          move.l    IntuiBase,a6    ; get Intuition
                                    ; ptr for library
                                    ; call
          lea       new_win(pc),a0  ; point to our
                                    ; NewWindow
                                    ; structure
          jsrlib    OpenWindow
          tst.l     d0
          beq       DosClos

          move.l    d0,our_win      ; save Window ptr
                                    ; for later
          move.l    d0,a0           ; keep Window
                                    ; pointer
          move.l    GfxBase,a6      ; get GraphicsLib
                                    ; ptr for library
                                    ; call

; We now find ourselves a pointer to the rastport we
; wish to use.  We need one of these for the graphics
; library calls wewish to make.
;
          move.l    wd_RPort(a0),a1
;
; We are now ready to plot a point on the screen
;
          moveq     #1,d0           ; the pen (color
                                    ; map entry) to use
          jsrlib    SetAPen         ; (NB. RPort in a1,
                                    ; Pen in d0,
                                    ; Lib ptr in a6)
          moveq     #10,d0          ; (x,y) coordinates
                                    ; of pixel in [d0,d1]
          moveq     #10,d1
          jsrlib    WritePixel      ; Draw the point

          move.l    DosBase,a6      ; get DosLib ptr
                                    ; for library call
          move.l    #250,d1         ; how long we want
                                    ; to wait
          jsrlib    Delay
;
; Now we close down and free our resources in an
; orderly manner.
;
          move.l    IntuiBase,a6    ; get Intuition ptr
                                    ; for library call
          move.l    our_win,a0      ; the window we
                                    ; want to close
          jsrlib    CloseWindow

          moveq     #0,d7           ; set up NO ERROR
                                    ; return
          move.l    SysBase,a6      ; get ptr to
                                    ; ExecBase (for
                                    ; CloseLibrary)

DosClos   move.l    DosBase,a1      ; point to library
                                    ; to close
          jsrlib    CloseLibrary

GfxClos   move.l    GfxBase,a1      ; point to library
                                    ; to close
          jsrlib    CloseLibrary

IntuClos  move.l    IntuiBase,a1    ; point to library
                                    ; to close
          jsrlib    CloseLibrary

AbortExit move.l    d7,d0           ; set up return
                                    ; code
          rts                       ; return to DOS

          cnop      0,4             ; align data to
                                    ; longword boundary
;
; Now we declare our data
;
new_win   dc.w      20          ; nw.LeftEdge = 20;
          dc.w      20          ; nw.TopEdge = 20;
          dc.w      300         ; nw.Width = 300;
          dc.w      100         ; nw.Height = 100;
          dc.b      0           ; nw.DetailPen = 0;
          dc.b      1           ; nw.BlockPen = 1;
          dc.l      0           ; nw.IDCMPFlags = NULL;
          dc.l      GIMMEZEROZERO ; nw.Flags
          dc.l      0           ; nw.FirstGadget = NULL;
          dc.l      0           ; nw.CheckMark = NULL;
          dc.l      Wttl        ; nw.Title = (UBYTE *)
                                ; "Set a point in here?";
          dc.l      0           ; nw.Screen = NULL;
          dc.l      0           ; nw.BitMap = NULL;
          dc.w      0           ; nw.MinWidth = 0;
          dc.w      0           ; nw.MinHeight = 0;
          dc.w      0           ; nw.MaxWidth = 0;
          dc.w      0           ; nw.MaxHeight = 0;
          dc.w      WBENCHSCREEN  ; nw.Type =
                                  ; WBENCHSCREEN;

Wttl      dc.b      'Set a point in here?',0

our_win   ds.l      1
DosBase   ds.l      1
GfxBase   ds.l      1
IntuiBase ds.l      1

dosname   dc.b      'dos.library',0
gfxname   dc.b      'graphics.library',0
intuname  dc.b      'intuition.library',0

          end
```

Finally some comments on the relative merits of C and Assembly language for the implementation of these programs. I will start by stating a personal preference for C and admitting to a considerable amount of bias. I am also a professional programmer, who daily works in a graphics related job involving both C and 68000 Assembler. I am thus a tyro at neither of these languages. However, to write the above C program from scratch and type it in and debug it took me aproximately 1.5 hours. The Assembler version by comparision took about 10 hours to type in and debug. This it should be noted was with a working C program as a model. I will admit that this as my first attempt at Amiga assembly language absorbed some time learning what not to do when using libraries, and thus 10 hours is longer than the real comparision value should be. I would guess that after reaching proficiency there would still be a factor of 3 or more in the time taken to develop similar applications in C and Assembler.

The times taken to process each language are also stangely askew. The Aztec C compiler Version 3.4a compiles the C program to an executable file in about 64 seconds. The MetaComco Assembler for the Amiga by comparision takes about 185 seconds to assemble (a much simpler operation???) the Assembler source. An interesting point of note is that the Aztec compiler produces an intermediate assembly file which it assembles with its own assembler.

	C	Assembler
Time to write & debug	1.5 hours	10 hours
Time to assemble/compile	64 secs	185 secs
Size of executable	~4000 bytes	~400 bytes

Finally, the sizes of the two executable files are also about a factor of 10 apart. There is no real noticable difference in the speed of execution of the two programs although on a floppy disk based system you might notice a difference in loading times. I have presented these thought in the small table above. Lastly for those without the necessary tools to acquire a copy or create their own, I have shown reproduced a sample of the output below.

Next month our regular SIG coordinator will be back. Until we see you at the next meeting, Share & Enjoy.

All You Ever Wanted to Know About the Amiga's File System but Were Afraid to Ask

by Eric Salter

When I started playing around with the Amiga's file system, I had to wade through 30,000 pages of literature and bumf to find out just what the hell I was doing. Naturally this is a very frustrating process and inevitably leads to a visit from an old friend telling you "Don't Panic" in large flashing, friendly, red lettering. I hope you have as much fun as I did. One thing I learned early on was to save frustration by setting up your programming environment to allow quick compiles (a large RAM disk). This is one thing you do a lot of when your experimenting - compiling, and if this is not too slow, then it almost, but never quite, makes up for the power you gain over the system with a compiled language. This article will present all examples in 'C'. The main reason for this is that I know 'C' reasonably well, but also, it is the language of the Amiga's libraries - e.g. Intuition which for the most part were written entirely in 'C'. This article will not teach 'C'. As of this writing, I have completed 10 articles on 'C' programming which I will be revising to cover the Amiga programming environment and will publish in this journal as they are revised.

Getting Started

I use the Lattice 3.10 C compiler. It doesn't matter which 'C' compiler you use as long as you are capable of driving the beast. If you can load and compile the following 'C' code without any compiler warnings or errors, you have not only successfully compiled a 'C' program, but you have used one of the three ways of accessing the Amiga's file system:

```
/* xyzzy.c */

#include <stdio.h>

main()
{
 printf("Nothing Happens\n");
}
```

The above code uses the high-level or stream I/O functions to accomplish its task. When we type **xyzzy** on the console, we should get the text: **Nothing Happens** (for the adventure fans). If you were not successful in compiling and running the above without errors, then go away and learn some 'C' programming before you tackle any more.

Three File Systems in One

When most people program in 'C', they do it on a UNIX environment, that is, their machine and its software runs under the UNIX operating system. As you are probably aware, 'C' and UNIX have a long association. UNIX was originally brought up on a DEC PDP-11 and was re-written in 'C'. Most of the utility programs on the UNIX O/S are written in 'C' and naturally, 'C' caters for the UNIX file-system. 'C' under UNIX offers two levels of dealing with files and devices - the original low-level, "unbuffered" UNIX-like system where function calls to deal with files (e.g. open, write, creat, close etc.) are in reality calls to the UNIX operating system itself. The other system, the high-level buffered or stream file-system, was created to isolate the programmer from the idiosyncrasies of the UNIX functions and the need to maintain your own buffers.

When we use 'C' on an operating system other than UNIX, the calls to the low-level UNIX-like file I/O functions are emulations of the UNIX operating system calls, usually by calls to the host's operating system. Luckily, under AmigaDOS, we are operating in an environment very close to UNIX and as a result, many programs that compile and run under UNIX can be imported, in most cases with little or no changes. Nevertheless, there is a group of file functions that are interfaces directly to the AmigaDOS system, and the UNIX-like system is built on these. Likewise, the stream file-system is built directly on the UNIX-like functions. To quote K&R, `On any particular system, the routines of the standard library have to be written in terms of the I/O facilities actually available on the host system.' Thus it is with 'C' on the Amiga.

An example of code using the three systems

In this example, we will use the three levels of interface to the file-system in 'C' on the Lattice 3.10 compiler environment. The code will send the string "Nothing Happens" to the user's current window.

```
/* xyzzy.c - stream I/O [high-level buffered I/O] */
#include "stdio.h"

void main()
{
 FILE *fp;

 /* ok, go open user's window.
  If fail exit with error code */

 if ((fp = fopen("*","w")) == NULL ) exit(FALSE);
 fprintf(fp,"Nothing Happens\n");
 fclose(fp);
 /* Always explicitly close files you open */
}
```

Our program uses the high-level stream I/O functions fopen, fprintf, fclose. These functions are not part of the Amiga's system software but are supplied by the compiler manufacturer in standard libraries - in Lattice this is lc.lib. These routines will use the Amiga's routines ultimately, but they isolate you from having to know that you are programming on a non-UNIX system. In this program, we define a file pointer, which is a pointer to a FILE structure. We then attempt to open the user's currently active window - the "*" under the AmigaDOS file system means the currently active window and can be treated as a file to which we can send a stream. We then send the string array "Nothing Happens" to the window and close our "handle" to it.


```
/* xyzzy2.c - using the UNIX-like file I/O */

#include <fcntl.h>

void main()
{
 int fd; /* known as the file descriptor but is
 really a file handle under AmigaDOS */

 if ((fd = open("*",O_WRONLY)) == EOSERR) exit(FALSE);

 /* Should inform user of file open fail but as we
 are attempting to open his current window for output
 we are rather stuck */

 write(fd,"Nothing Happens\n",16);
 close(fd);
}
```

In this example using the UNIX-like file I/O, we deal with a file with a file descriptor "fd". On a UNIX system file descriptor 0 refers to the standard input - stdin, 1 refers to standard output - stdout, and 2 refers to standard error - stderr. Lattice in its wisdom has decided this is not for its own compilers and under version 3.10 of the compiler, the fd returned from a call to open() is the AmigaDOS file handle - this is plainly wrong and Lattice should be castrated. Even though the UNIX I/O is not defined under the proposed ANSI standard C, it is no reason to do something totally non-standard. Anyway, the window is sent the text but notice we have to count for ourselves how many characters are in the string - we could use the function strlen() to calculate it for us though and this is encouraged. Ohbytheway, Lattice calls strlen() strnlen - again non-standard.

Lowest level interface to AmigaDOS

```
/* xyzzy3.c - level 0 file I/O */

#include "libraries/dos.h"

void main()
{
 int fh;

 if ((fh = Open("*",MODE_OLDFILE)) == 0) exit(FALSE);
 Write(fh,"Nothing Happens\n",16);
}
```

stdin, stdout and stderr under Lattice

We mentioned above that the usual way of specifying stdin,stdout and stderr with the file descriptors 0,1 and 2 respectively, doesn't work in Lattice 3.10 or above. Well how do you specify in the level 1 file I/O (UNIX-like I/O) that you want to send something to stdout or get something from stdin? Where are the file handles for the default input and output streams? Well, Lattice doesn't make it easy for you. These file handles refer to the files attached to the input and output of your program. If you run a program from the CLI, the input and output file handles will refer to your console. If on the other hand, you invoked a program with the re-direction operators > and < e.g. cat < inputfile > outputfile, AmigaDOS will attempt to open these two files "inputfile" and "outputfile" and these will be used for input and output for your program. The file handles will be for these files rather than the console. Under AmigaDOS, you get the input, and output file handles of your program via calls to the AmigaDOS functions Input() and Output().

```
 int _infh;
 int _outfh;

 _infh = Input();
 _outfh = Output();
```

Therefore, using the above to send our message to the standard output:

```
void main()
{
 int _outfh;

 _outfh = Output();
 write(_outfh,"Nothing Happens\n",16);
}
```

Note that upper and lower case matter. There is a function write() belonging to the lc.lib and a function Write() which is the low-level AmigaDOS function. It wouldn't make any difference here if we used Write instead of write because both functions expect the same arguments - what then is the difference? Well, the Lattice write() maintains error checking and abort checking code and maintains some buffers which are used by the high-level stream I/O. write() will eventually call Write() [actually via a function called dwrite() - but this complicates the issue]. In the above code, we could have obtained the output file handle from the structures that Lattice set up for us in the module _main(). _main() is called by the startup code which in turn calls your code at main(). Within _main(), the input filehandle is found in stdin->_file, the output filehandle is in stdout->_file and the standard error file handle is set up to be the file handle of your currently activated window using the code we had above: Open("*",MODE_OLDFILE) and is specified by stderr->_file. There is a trap here. If you start doing fancy things like calling the entry point to your routine _main(), to save unwanted code being added to your run-time size the above example will not work, indeed, all code requiring high level I/O and command line argument passing will fail, unless you do it yourself in your own _main().

You can see life is fairly complicated using Lattice. This is partly Lattice's fault but partly the complexity of the Amiga system - another level of abstraction with file descriptors means more overhead in terms of system memory and execution speed. Actually, using AmigaDOS is easy, doing the same things with Intuition and message passing is hell. We may, if I don't go mad first, get there sometime.

==

Public Domain Update

Almost all of the latest Fred Fish disks contain entries from the Badge Killer Demo contest, recently held to try to find the Amiga demo. I haven't seen all of them yet, since I haven't got enough memory to run some of them, but I particularly like Leo's effort on disk 115, Marketroid.

Fish Disk #111

AmyLoad — A graphical monitor of cpu, blitter, and memory use. Includes two components; load.device, which monitors system parameters, and amyload, which is the user interface and display program. Includes source.

AssignDev — Assigns multiple names to a given device. For example, allows the names "df0:" and "df3:" to refer to the same physical device. This is a modified version of the original released on disk number 79. Includes source.

Gauge — Continuously displays memory usage in a vertical bar graph, similar to the workbench "fuel gauge" type display for disk space. Binary only.

HeliosMouse — Another "sunmouse" type program. Automatically activates a window simply by moving the mouse pointer into the window. Version 1.1, an update to the version released on disk 94. Includes source.

Labels — Alphabetic and numeric ordered cross reference lists of defined system constants. Recommended for debugging purposes only, use the symbolic values in programs!

Mandel — Another mandelbrot generator program, with bits and pieces of code from C. Heath and R.J. Mical. Includes source.

PopLife — A PopCLI type thingie that instead plays life all over your screen. Lots of bits and pieces from Tomas Rokicki's blitlab and John Toebes' PopCLI. Includes source.

Fish Disk #112

BeachBirds — Jerrold Tunnell's entry to the Badge Killer Demo Contest. Uses sprites and sound to portray a beach scene. Runs on a 512K machine. Binary only.

Bully — Mike Meyer's entry for the Badge Killer Demo Contest. Pushes all open screens around (thus the name "bully"). Designed for showing off more than one demo at a time Includes source.

DropShadow — Dropshadow version 2, rev 0, for use with Bryce Nesbitt's Wavebench demo. Binary only.

HagenDemos — Joel Hagen's Badge Killer Demo contest entries, "RGB" and "Focus". RGB was the overall winner of the contest. It requires a one meg Amiga to run. Binary only.

Viacom — Latest version of viacom for use in conjuction with the WaveBench demo. Binary only.

WaveBench — This is Bryce Nesbitt's Badge Killer Demo Contest entry. It is a neat screen hack, and runs on 512K machines. For more laughs, try in conjunction with Viacom or Ds (Dropshadow). Includes source.

Fish Disk #113

AmiCron — A simple Unix "cron" type program, which is a background task that uses a disk-resident table to automatically run certain tasks on a regular basis, at specific times. Version 2.3, includes source.

Dme — Version 1.28f of Matt's text editor. Dme is a simple WYSIWYG editor designed for programmers. It is not a WYSIWYG word processor in the traditional sense. Features include arbitrary key mapping, fast scrolling, title-line statistics multiple windows, and ability to iconify windows. Update to version on disk number 93, includes source.

DosDev — Example DOS device driver in Manx C. Version 1.10, includes source.

M2Amiga — Demo version of the final product M2Amiga. A fast single pass Modula-2 compiler with editor, linker, a small set of interface and standard libraries. Compiles only small demo programs by limiting codesize and imports. Further development of the ETHZ compiler on Disk 24. Binary only. Demos with source.

NoIconPos — This program clears the position info of any of your icons to allow WorkBench to pick a reasonable place for the icon again. Useful for disk and drawer icons where Snapshot rewrites the icon and the window information. Written in Modula-2, another demo for M2Amiga, showing the simplicity of programming with this Modula-2 compiler.

Fish Disk #114

CDecl — English to C (and vice versa) translator for C declarations. This little gem will translate english such as "declare foo as pointer to function returning pointer to array 10 of pointer to long" into "long *(*(*foo)())[10]", and vice versa. An absolute must for anyone except possibly the most hardcore C guru. Includes source.

Vt100 — Version 2.7 of Dave's vt100 terminal emulator with kermit and xmodem file transfer. Includes a few bug fixes posted to Usenet shortly after the posting of version 2.7. This is an update to the version released on disk 55. Includes source.

WBLander — This entry from the Badge Killer Demo Contest is a special version of the WBLander program from disk 100. The ending is unique. Also uses sound effectively. Includes source.

Fish Disk #115

Killer — Killer is an incredible demo written by Robert Wilt. It won fourth place in the Badge Killer Demo Contest. Requires at least one meg of memory to run. Sound is also an important part of the demo so be sure to turn it up. Binary only.

Marketroid — Marketroid is Leo's entry for the Badge Killer Demo Contest. It is another devious sprite oriented demo with lots of "in" jokes. Runs on a 512K machine. Includes source.

Fish Disk #116

Movies — A ram animation system with three different example animations; Kahnankas, Rocker, and F-15. Kahnankas won a close second in the Badge Killer Demo Contest. Both Kahnankas and Rocker run on a 512K Amiga and show off overscan HAM mode. Includes a animation player program (movie), animation builder programs (dilbm, pilbm), and a text/graphics display program (vilbm).

Fish Disk #117

AMUC_Demo — A really neat horizontal scrolling demo that is a 2400 x 200 pixel 32 color IFF picture composed of digiview snapshots of members of the AMiga Users of Calgary, superimposed on a very wide picture of the Calgary Skyline. Binary only.

ExP_Demo — Demo version of Express Paint 1.1. This is the program that was used to create the huge scrolling demo picture in the AMUC_Demo drawer on this same disk. Binary only.

Fish Disk #118

Empire — This is a complete rewrite, from the ground up, in Draco, of Peter Langston's Empire game. Empire is a multiplayer game of exploration, economics, war, etc, which can last a couple of months. Can be played either on the local keyboard or remotely through a modem. This is version 1.0, shareware, and includes source code.

HAMmmm — This is Phil's entry for the Badge Killer Demo Contest. HAMmmm displays lines whose end points are bouncing around the screen, which is a double buffered HAM screen. The Y positions of the points are continuously copied into an audio waveform that is played on all four channels, and the pitch of a just intoned chord is derived from the average X position of these points. Includes source in JForth.

Stars — Hobie's entry for the Badge Killer Demo Contest. Based on original code by Leo Schwab, has credits longer than the actual demo. Runs on 512K Amiga. Binary only.

WireDemo — Matt's entry for the Badge Killer Demo Contest. Demonstrates the Amiga's line drawing speed. Runs on a 512K Amiga. Includes source.

==

Printers and Your Amiga
by Con Kolivas

If you are like me, you have probably updated from your C64 to a species of Amiga. If so, you probably also have an Epson GX80 printer to your name. In the transition from C64 to Amiga, you no doubt realised that you needed a lot more than your GX80 offered. Well, before you put the boot into your printer or trade it in for some other extremely expensive, overpriced machine, you should read on.

It will probably come as a surprise to you to find out that your GX80 is really an LX80 in disguise. The only thing that separates the two is one small chip. If you get put off by electronics, don't worry, you don't need to unscrew or solder anything. When you purchased your GX80, what you got was a powerful machine downgraded by the simple interface plugging it into whatever your computer was - it was Epson's way of supplying us "small fry" with an Epson (Oh, wow, look at the brand!) The interface itself contains a chip, but not the chip.

If you look at the plug which fits into the printer, you might be shocked to find about half the pins unconnected. If you can remember back to when you purchased it, you should remember the cigarette-paper thin size of the instruction manual. For just $75 (and another $30 for the cable), you can turn your GX80 into an LX80. What you need is Epson's printer interface cartridge #8620 which will give you a parallel input into your printer. The cartridge is not easy to find, but Maxwell's in Abbotsford have them.

When you open the box, gee whizz - a big manual! Looking through the manual, you will find an extra font (elite), italics, underline, super/subscripts, seven different graphics densities, 9 pin graphics and so on. So, there you have it. If you've already sold your GX80, then bad luck. Otherwise, go get this interface, set preferences to Epson, and away you go.

Now, if you could see how this article was written, you would notice that it was by pen and paper, which says something considering this is an article about printers. If you're like me in that you don't have any of the big gun programs yet, then you might also be writing in pen. More to the point, if you have tried using the small, cheaper programs, you might find that the escape codes in your printer's instruction manual don't work. The reason for this is that the Amiga has its own set of escape code sequences that you will find in the back of your instruction manual.

What the Amiga does when you use output to PRT: or use BASIC's LPRINT etc, is to look up its preferences table, then the printer driver, and convert these Amiga codes into the appropriate sequences for your printer. (If you got lost in that description, don't worry.) What this means to you is that if you want to use your printer's facilities, you must use the escape sequences listed in your Amiga manual, not your printer manual. Not all of the listed codes work for all printers, and you may find that many options are simple not supported.

One way to get around this is (in BASIC) is to open a file using the PAR: or SER: printer devices. For example, OPEN "PAR:" FOR OUTPUT AS #2, then PRINT #2, data. Then, in place of LPRINT, use PRINT #2. This bypasses the printer driver conversions, and transmits exactly what you send your printer. (See the chapter on working with files and devices in your BASIC manual for more details.)

If you are more experienced in C programming than most people, you may like to write your own printer driver. Amigan disk #10 has a program that will do this for you. If you wish to write your own, you will need the "Amiga ROM Kernel Manual of Libraries and Devices" to do this.

Thank you for reading my article, and I hope I have a damn word processor if I ever want to write another.

==

Editor's Column

Written 03-Jan-88

Happy New Year, and happy 200th to Australia. The club received two Christmas cards this year, one from Commodore, signed by Craig Tegel, and one from Snap Instant Printing, who do the newsletters. Thanks.

During the last few weeks, I've dug deep into Amigaland and found a few interesting items. First is an adaptor that lets you use an IBM hard disk drive and controller on your Amiga. Called "The Wedge", this item is being distributed through an Amiga group in Canada. I've ordered one, and I'll let you know all the details when it arrives. I've also asked whether AUG could distribute these boards in Australia, so cross your fingers for real cheap hard disk drives (I'm talking under $1000 for 20 megs!). Also in Canada, I've tracked down a guy who is making speed-up kits for the Amiga. Selling for about $100 cdn, they let the 68000 processor run at 14.2Mhz, twice the normal speed. I'm waiting on further details, and again, I'll let you know as soon as I can.

Have you heard of ARP? For those who haven't, ARP is the AmigaDOS Replacement Project, a Bix-based project spearheaded by Charlie Heath (author of Tx-Ed) to write C and Assembler replacements for the AmigaDOS commands we all know and hate. ARP V1.0 is now available, and includes new versions of 15 or so AmigaDOS commands, as well as arp.library, which contains common code used by all these commands. The commands are both smaller and have more functionality than their original counterparts. AUG has ordered the ARP distribution disk, and it will be available from the club disk library soon.

I've also found out about, and subscribed to, Amigamail, a publication done by Commodore USA. This will be available from our library as soon as it arrives. There are also a few other interesting items I'm still in the process of finding out about, stay tuned for more details.

I've had to do the newsletter twice this month, once with WordPerfect, and again, **properly** the old way. I think I've been fair to WordPerfect by giving it a really good go, but I just don't think it is suitable for doing this newsletter. I find I am constantly fighting with it, trying to convince it to do what I need it to do. For instance, having a spelling checker is a great idea, but if it takes about an hour to check this text, I can do without it. The whole thing is just too slow at doing what I need it do to, and far from "What you see is what you get", I found I had to print the whole newsletter several times before I got it right.

Unfortunately, I seem to have found that WordPerfect is just not the right tool for the job I've got to do. It reminds me of a Swiss army knife, with just about everything you're ever likely to need for almost anything. Still, even with all those tools, a Swiss army knife just isn't suited to house building, is it....

So, I think we'll have to re-sell our copy, bite the bullet and buy a laser printer and a "desktop publishing" program. If you'd like to buy the club's WordPerfect at a very good price, ring me at the number in front of the newsletter. If we get a laser printer, we'll also have a near-new Brother HR-40 daisy wheel printer, worth about $1750 new. Maybe we could do a package deal for about $1800 or so for both. That'd pay for about half of the laser printer!

Also on the down side, the club's Introduction to the Amiga video tape seems to have disappeared. Whoever has it must have finished with it by now, so do your fellow club members a favour and return it. The same goes for any overdue library items that anyone has. The library people tell me that there are a few **very** overdue items. These things are **club** resources, please send them back.

You may have noticed that the newsletter is slightly late this month - that's because you lot didn't contribute enough (notice that we have 4 less pages), and those contributions that did turn up were delayed until after Christmas in the mail. If not for the Fred Fish disks turning up, we mightn't have had enough for a newsletter at all. The reason that I didn't expect any Fish disks this month was that the last bank draft we sent Fred (for $200 US) was returned by his bank as uncollectable. It turns out that the bloody State Bank (AUG's bank) here filled out the draft on the wrong form. So, Fred had to send the cheque back, we had to get a new one, and then post that back to him. With all that stuffing around, I didn't expect new disks for a month or so at least. I'd like to thank Fred for sending the latest disks out on the same day he received our new draft. Thanks.

Don't forget that there is no normal January meeting, instead come to our Barbeque, from noon onwards on Saturday, January 16th. In complete contrast to where we now hold our normal meetings (ie Monash in Clayton), the barbeque is right over the other side of town in Bundoora. (50Km from where I live!). So, the members on that side of town should have no troubles at all in getting there. AUG will supply cold soft drinks and money for the BBQ, you supply your food and cutlery, etc.

Next real meeting will be on the second Saturday in February (the 13th), at Monash again. Whilst we've had a few complaints, no-one has been able to suggest a suitable alternative venue. Think about it, and don't be afraid to suggest anything.

Who Are We?

The Amiga Users Group is a non-profit association of people interested in the Amiga computer and related topics. With almost 900 members, we are the largest independent association of Amiga users in Australia.

Club Meetings

Club meetings are held at 2pm on the second Saturday of each month in the Rotunda at Monash University, Wellington Road, Clayton. Details on how to get there are on the back cover of this newsletter. The dates of upcoming meetings are:

Saturday, February 13th at 2pm
Saturday, March 12th at 2pm
Saturday, April 9th at 2pm

Production Credits

This month's newsletter was edited by Peter Jetson. Equipment and software used was: TurboDOS S-100 computer, Brother HR-40 printer, Gemini 10x printer, Wordstar, Fancy Font and Grabbit. I tried to use WordPerfect on the Amiga, but failed.

Copyright and Reprint Privileges

Contributions

Articles, papers, letters, drawings and cartoons are actively sought for publication in Amiga Workbench. Please submit your contributions on disk, since that means they don't have to be re-typed! All disks will be returned! Please save your article in **text-only** format (If it can be loaded by ED, it is text-only). Absolute deadline for articles is 16 days before the meeting date. Contributions can be sent to: The Editor, AUG, PO Box 48, Boronia, 3155.

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 48, Boronia, 3155

Public Domain Software

Disks from our public domain library are available on quality 3.5" disks for $8 each including postage on AUG supplied disks, or $2 each on your own disks. The group currently holds over 148 volumes, mostly sourced from the USA, with more on the way each month. Details of latest releases are printed in this newsletter, and a catalog disk is available.

Member's Discounts

The **Amiga Users Group** negotiates discounts for its members on hardware, software and books.

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!

Back Issues of Newsletter

All back issues of Amiga Workbench are now available, for $2 each including postage. Back Issues are also available at meetings.

AmigaLink - Our Bulletin Board System

The Amiga Users Group operates a bulletin board system devoted to the Amiga, using the Opus message and conferencing system. AmigaLink is available 24 hours a day on (03) 792 3918, and can be accessed at V21 (300bps), V22 (1200bps) or V23 (1200/75bps), using 8 data bits, 1 stop bit and no parity.

AmigaLink is part of the world-wide Fido/Opus network of bulletin boards, and we participate in the national and international Amiga conferences. AmigaLink has selected Public Domain software available for downloading, and encourages the uploading of useful public domain programs from its users. AmigaLink is FidoNet node number 631/324.

Newsletter Advertising

The Amiga Users Group accepts commercial advertising in Amiga Workbench subject to the availability of space at these rates:

Quarter page	$20
Half page	$40
Full page	$70
Double page spread	$120

These rates are for full-size camera-ready copy **only.** We have no photographic or typesetting facilities. Absolute deadline for copy is 16 days before the meeting date. Send the copy and your cheque to: The Editor, AUG, PO Box 48, Boronia, 3155, Victoria.

Amiga Users Group Committee

Co-ordinator:	Bob Scarfe	376 4143 Kensington
Vice Co-ord:	Fergus Bailey	211 7845 Malvern
Meeting Chair:	Ron Wail	878 8428 Blackburn
Secretary:	John Elston	375 4142 M' Ponds
Treasurer:	(temporarily vacant)	
Membership:	Neil Murray	794 5683 Dandenong
Purchasing:	Bohdan Ferens	792 1138 Dandenong
Book Library:	Joan Wood	580 7463 Aspendale
Disk Library:	Geoff Sheil	578 8362 Brighton
	Margaret Bedson	578 8362 Brighton
Editor:	Peter Jetson	762 1386 Boronia
SMAUG Co-ord:	Roland Seidel	890 3934 Box Hill

Newsletter Back Issue Order Form

Issue Numbers:									
Number of issues at $2 each								$	
Less 10% for 5 or more								- $	
Club Use Only								Total $	
Mail to: Amiga Users Group, PO Box 48, Boronia, 3155									
Member's Name:									
Address:									

SOFTWARE ORDER FORM

Disk numbers :									
Disks supplied by Amiga User Group @ $8								$	
Disks supplied by member @ $2								$	
Club Use Only Receipt #: Mailed on: / /								Total $	
Mail to: Amiga Users Group, PO Box 48, Boronia, 3155, Victoria.									
Member's Name:									
Address:									

Application for membership of The Amiga Users Group Inc

Membership is $20 per year. Send your cheque to: **Amiga Users Group Inc, PO Box 48, Boronia, 3155**

Surname: ____________________

First name: ____________________ (no initials)

Address: ____________________

____________________ Postcode: ________

Phone Number: ____________ STD Code: ______

What services would you like AUG to provide: ________

____

____

Signed: ____________ Date: ________

Details on this side are optional

Year of birth: ________ Which model Amiga: ________

Occupation: ____________________

Interests: ____________________

____

____

Dealer's Name: ____________________

Dealer's Address: ____________________

____________________ Postcode: ________

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.

Club Use Only	Date	Paid	Rcpt #	Memb #	Card Sent