AmigaProgFAQ

COLLABORATORS			
	TITLE : AmigaProgFAQ		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		November 29, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaProgFAQ

1.1 main

```
Please address feedback to: jbickers@templar.actrix.gen.nz. Last
modified 21st May 1993. The information is sourced from Usenet,
but the mistakes are mine.

If you post an article that you think is an answer to a frequently
asked question, please consider adding the string "FAQ" to the
article somewhere. This would make it easier for me to pick up
such answers.


TABLE OF CONTENTS

1   MAILING LISTS
2   LIBRARIES
3   AVAILABLE COMPILERS, DOCUMENTATION & TOOLS
4   COMMUNICATIONS INFORMATION
5   DOS INFORMATION
6   EXEC INFORMATION
7   GRAPHICS INFORMATION
8   INTUITION INFORMATION
9   SAS C INFORMATION
```

1.2 maillists

```
*** 1   MAILING LISTS

    A number of mailing lists are set up for email discussion of
    specific subjects. These include:

    a68k-request@castrov.cuc.ab.ca
        Discussion of Charlie Gibb's A68k freeware assembler.

    amiga3d@ahnold.cs.umass.edu
        Support for 2D and 3D Amiga graphics programming. Send
        requests to kiniry@cs.umass.edu or kiniry@ahnold.cs.umass.edu.
```

```
    amigalisp-request@contessa.palo-alto.ca.us
        Discussion of LISP, Scheme, ML, Gofer, functional programming,
        and other such things.

    amos-request@access.digex.com
        Discussion of AMOS programming. Subscribe by sending a message
        to this address with a line in the body of the message like:
            SUBSCRIBE <address>
        For example, "SUBSCRIBE jbickers@templar.actrix.gen.nz". Post
        to amos-list@access.digex.com.

    dice-request@castrov.cuc.ab.ca
        Discussion of Matt Dillon's shareware DICE C compiler.

    xpr-request@aldhfn.akron.oh.us
        Discussion of the XPR file transfer protocol.
```

1.3 libraries

```
*** 2   LIBRARIES

    disassemble.library [Chris Gray]
        Routines to do controlled symbolic disassembly of 68000, 020,
        882 and 851 instructions. It is available on Fisk disk #240,
        and includes a disassembler 'dis'.

    iff.library [Christian Weber]
        Routines to read and write IFF files, with direct support for
        ILBM and ANIM forms for pictures and animations. ANIM support
        includes decoding of DLTA chunks. It is also XPK-aware. It is
        available from amiga.physik.unizh.ch and on Fish disk #674.

    pattern.library [Angela Schmidt]
        AmigaDOS pattern matching routines which are less buggy than
        OS (KS2.0) or ARP (KS1.2) routines, and include some special-
        case optimizations. It is available from ftp.wustl.edu and on
        Fish disk #625.

    reqtools.library [Nico Frangois]
        String, screen, color, file and font requesters, with a 2.0
        look. It is available from amiga.physik.unizh.ch.

    xpkmaster.library [Dominik Mueller et al]
        A variety of data compression and encryption routines managed
        through a set of sublibraries. Can compress RAM to RAM, file
        to RAM, RAM to file, etc. The user and development archives
        are available from amiga.physik.unizh.ch.

    xpr<proto>.library [Willy Langeveld et al]
        A variety of external protocol libraries that meet the XPR 2.0
        specification. The spec. can be found on Fish disk #247. The
        libraries themselves are often distributed independently. Join
        the XPR mailing list for information about XPR 3.0.
```

1.4 compilers

```
*** 3   AVAILABLE COMPILERS, DOCUMENTATION & TOOLS

    a68k
        Charlie Gibbs' freeware 680x0 assember, available from
        amiga.physik.unizh.ch.

    ACE 1.02
        David Benn's freeware AmigaBASIC compiler, available from
        amiga.physik.unizh.ch as ace102.lha.

    AdaEd
        An Ada implementation, available from wuarchive.wustl.edu as
        programming/programming/ada/adaed-1.0.11a.lzh.

    AmigaGuide
        CBM's on-line documentation system, pretty much freeware but
        check the license document for specifics. Available from the
        Aminet sites.

    AMOS
        Francois Lionet's programming language.

        Email: lionet@lionfl.adsp.sub.org
        Post.: Europress,
               Europa House, Adlington Park, Macclesfield SK10 4NP
        Voice: 0625 859333

    ARexx
        William Hawes' commercial REXX package. The address is:

        Post.: Wishful Thinking Development Corp.,
               PO Box 308, Maynard, MA 01754, USA.
        Voice: +1-508-568-8695

    AutoDocs
        See "Native Developers Update Kit" below.

    C Manual 3.0
        Anders Bjerin's collection of C documentation and examples for
        Amiga programming. Available on Fish disks 691 through to 695.

    Comeau C++ 3.0 With Templates
        Comeau Computing's commercial C++ package. Their address is:

        Email: attmail.com!csanta!comeau
        Post.: Comeau Computing,
               91-34 120th Street, Richmond Hill, NY 11418, USA.
        Voice: +1-718-945-0009 / Fax: +1-718-441-2310.

    DICE
        Matt Dillon's shareware C compiler, available from
        amiga.physik.unizh.ch. His address is:

        Email: dillon@overload.berkeley.ca.us
```

Post.: Matthew Dillon,
1005 Apollo Way, Incline Village, NV 89451, USA.

Draco
Chris Gray's Draco language. It is a general programming language like C, but with tidier syntax and more protective semantics. Package includes full V1.3 headers, compiler, linker, screen editor, run-time system, examples, docs, etc. It is available on Fish disks #77 and #201.

GadToolsBox
Gadget design tool, available from the Aminet sites.

Gofer
See the LISP entry for more information.

GNU C
Markus Wild's port of this freeware C compiler, available from amiga.physik.unizh.ch. Includes C++ capability.

JForth Professional V3
Delta Research's commercial Forth package. Their address is:

Email: phil@ntg.com (Phil Burk), haas@starnine.com (Mike Haas)
Post.: Delta Research,
P.O.Box 151051, San Rafael, CA 94915-1051, USA.
Voice: +1-415-461-1442

LISP
Several implementations are available from gatekeeper.pa.dec.com, in the directory /pub/micro/amiga/lisp. The file LISP.LIST has pointers to other sources. An up-to-date version of this is available via BMS from contessa.palo-alto.ca.us as bms:pub/lisp.list. See the Amiga LISP mailing list for more information.

ML
See the LISP entry for more information.

Native Developers Update Kit
CBM's machine-readable documentation, with the AutoDocs, the latest #include files, and the Software ToolKit II. Available for approximately 20 USD + postage from:

Commodore Business Machines, Inc.
Department C
1200 Wilson Drive, West Chester, PA 19380, USA

or: Hirsch & Wolf oHG
Attn: Hans-Helmut Hirsch
Mittelstr. 33, 5450 Neuwied 1, Germany

Voice: +49-2631-24485 / Fax: +49-2631-23878

PCQ Pascal
Patrick Quaid's freeware Pascal compiler, available from amiga.physik.unizh.ch.

RCS 5.6
Freeware Revision Control System, available on AmiNet sites in dev/misc/hwgrcs.lha and dev/misc/hwgrcssrc.lha.

Scheme
Suitable Schemes for use with Abelson & Sussman's "Structure and Interpretation of Computer Programs" are SIOD, Ed Turner's Scheme, or XScheme. If you have experience to the contrary, please let amigalisp@contessa.palo-alto.ca.us and the FAQ list maintainer know. These systems are available at gatekeeper.pa.dec.com, in /pub/micro/amiga/lisp. SIOD is also on Fish disk #525, and Scheme is on disk #149.

See the LISP entry for more information.

1.5 comm

*** 4 COMMUNICATIONS INFORMATION

4.1 Dropping DTR.

a. If you are using a CBM device, eg serial.device, you must call CloseDevice() to drop DTR. DTR is asserted when you open the device, and no other system mechanism is provided to drop it. Be aware that this fails if another application has opened the device in SHARED mode.

b. Many other devices support an extra command to control the DTR and RTS lines. This was originated by ASDG. The following is extracted from code originally posted by Russell McOrmond:

```
#define SIOCMD_SETCTRLLINES 0x10
#define SIOB_RTSB  0
#define SIOB_DTRB  1
#define SIOB_RTSF  (1<<SIOB_RTSB)
#define SIOB_DTRF  (1<<SIOB_DTRB)

IOSer.io_Command = SIOCMD_SETCTRLLINES;
IOSer.io_Offset  = SIOB_DTRF;
if (raising DTR) IOSer.io_Length = SIOB_DTRF;
else IOSer.io_Length = 0;
```

The io_Offset is a mask of bits to be affected (RTS and DTR), and the io_Length is a value to set each bit to. So to drop DTR you set the DTR bit in io_Offset, and clear the DTR bit in io_Length. To raise DTR again, you set the DTR bit in both io_Offset and io_Length.

c. If you are using the internal serial port with CBM's serial.device, you can drop DTR without calling CloseDevice by banging the hardware. Again the following is extracted from code originally posted by Russell McOrmond:

```
struct CIA *Ciab = (struct CIA *)0xbfd000;
```

```
    Disable();
    Ciab->ciaddra |= CIAF_COMDTR;                   /* Set DTR as output */
    if (raising DTR) Ciab->ciapra &= ~CIAF_COMDTR;  /* Raise */
    else Ciab->ciapra |= CIAF_COMDTR;               /* Drop */
    Enable();
```

1.6 dos

```
*** 5   DOS INFORMATION
    5.1     Can an ACTION_READ to a console be aborted?
    5.2     Need a pointer to a CLI window.
    5.3     Checksum errors, trimnews and hard links.
    5.4     Scanning each entry in a multi-assign.
    5.5     Read character from console immediately (raw).
    5.6     How the Amiga FS works.
    5.7     AmigaDOS version command.
    5.8     Determining program path.
```

1.7 action_read

```
    5.1     Can an ACTION_READ to a console be aborted?

    No, with the exception that if you close the console, then all
    pending packets are returned.
```

1.8 cli

```
    5.2     Need a pointer to a CLI window.

    The following code is SAS C specific, but should be easy to modify
    for other compilers. The essential thing is that you have to send
    an ACTION_DISK_INFO packet to the console handler for the CLI.

    /* findwindow.c - utility routine to find window of a CLI.
        From: deven@rpi.edu (Deven T. Corzine)
        Subject: Re: Finding Windows
        Date: 20 Jul 90 16:14:05 GMT
     */
    #include <exec/types.h>
    #include <exec/memory.h>
    #include <proto/exec.h>
    #include <proto/dos.h>

    struct Window __regargs *FindWindow()
    {
       register struct DosLibrary *DOSBase;
       register struct Window *win;
       register struct Process *proc;
       register struct CommandLineInterface *cli;
```

```
   register struct InfoData *id;
   register struct StandardPacket *pkt;
   register struct FileHandle *fh;
   register BPTR file;
   register long ret1,ret2;

   if (DOSBase=(struct DosLibrary *) OpenLibrary(DOSNAME,0)) {
      if (id=(struct InfoData *)
        AllocMem(sizeof(struct InfoData),MEMF_PUBLIC|MEMF_CLEAR)) {
         if (pkt=(struct StandardPacket *)
           AllocMem(sizeof(struct StandardPacket),MEMF_PUBLIC|MEMF_CLEAR)) {
            proc=(struct Process *) FindTask(NULL);
            if (cli=(struct CommandLineInterface *) (proc->pr_CLI<<2)) {
               ret1=cli->cli_ReturnCode;
               ret2=cli->cli_Result2;
               if (file=Open("*",MODE_NEWFILE)) {
                  if (IsInteractive(file)) {
                     pkt->sp_Msg.mn_Node.ln_Name=(char *) &(pkt->sp_Pkt);
                     pkt->sp_Pkt.dp_Link=&(pkt->sp_Msg);
                     pkt->sp_Pkt.dp_Port=&(proc->pr_MsgPort);
                     pkt->sp_Pkt.dp_Type=ACTION_DISK_INFO;
                     pkt->sp_Pkt.dp_Arg1=((ULONG) id)>>2;
                     fh=(struct FileHandle *) (file<<2);
                     PutMsg(fh->fh_Type,(struct Message *) pkt);
                     WaitPort(&(proc->pr_MsgPort));
                     GetMsg(&(proc->pr_MsgPort));
                     win=(struct Window *) id->id_VolumeNode;
                  }
                  Close(file);
               }
               cli->cli_Result2=ret2;
               cli->cli_ReturnCode=ret1;
            }
            FreeMem(pkt,sizeof(struct StandardPacket));
         }
         FreeMem(id,sizeof(struct InfoData));
      }
      CloseLibrary((struct Library *) DOSBase);
   }
   return(win);
}
```

1.9 checksum

```
5.3     Checksum errors, trimnews and hard links.
        [Randell Jesup]

The filesystem has a problem with deleting the target of a
hardlink, especially on dir-cached partitions (DOS\5), but also
hardlinked directories on FFS, and on files and directories under
OFS (though you may not see the problem unless you need to recover
the partition).

Temporary solutions: don't use hardlinks to directories, and don't
put UUCP news on DCFS partitions.  This will be fixed in a future
```

release of the FS.

1.10 multiassign

5.4 Scanning each entry in a multi-assign.
[Randell Jesup]

```
BOOL do_something_to_all(char *path, BOOL (*function)(BPTR lock))
{
struct DeviceProc *dp = NULL;
struct MsgPort *old_fsport;
BPTR lock, old_curdir;
char *remainder;
LONG err;

    // NOTE: not strrchr - PathMan has files with ':'s in them
    remainder = strchr(path,':');
    if (remainder == NULL)
        remainder = path;
    else
        remainder++;            /* point past ':' */

    while (1) {
        dp = GetDeviceProc(path,dp);
        if (!dp)
        {           /* getdevproc freed dp for us */
                    // NOTE: 2.04 and 3.0 have a bug, and never return
                    // ERROR_NO_MORE_ENTRIES.  Accept 0 as no error as well
                    // This will be fixed next release.
            err = IoErr();
            if (err == 0 || err == ERROR_NO_MORE_ENTRIES)
                return TRUE;        // all done
            else
                return FALSE;       // never found anything
        }
        /* save filesystem port pointer, set default FS to target */
        old_fsport = SetFileSysTask(dp->dvp_Port);
        old_curdir = CurrentDir(dp->dvp_Lock);      // may be NULL

        /* we have an entry, get a lock on the remainder of path */
        lock = Lock(remainder,SHARED_LOCK);

        /* reset filesystem port and current dir */
        (void) SetFileSysTask(old_fsport);
        (void) CurrentDir(old_curdir);

        /* we got a lock on it, call user function.  Function can */
        /* return FALSE to stop the scan. */
        if (!lock || !(*function)(lock))
        {
            UnLock(lock);           // NULL is safe
            FreeDeviceProc(dp);
            return FALSE;
        }
        UnLock(lock);
```

```
    }
}
```

Warning: that was written off the top of my head, but I do have the source code for reference. Also, at least one developer has used this and it works.

1.11 raw

5.5 Read character from console immediately (raw).

It is often useful to have functions like getchar() return immediately when the user presses a key, rather than waiting until they hit Enter. To do this, you need to switch your console into "raw" mode. The line-based mode is referred to as "cooked" mode.

a. Under KS2.0, dos.library has a function SetMode() that does it all.

b. Under DICE, the link library function setvbuf() also switches modes automatically.

c. Otherwise, you need to send an ACTION_SCREEN_MODE packet to the console yourself. Set arg[0] to -1L for raw mode, 0 for cooked mode.

1.12 amigafs

5.6 How the Amiga FS works.

[Randell Jesup, on coroutines in Amiga filesystems]

Yes, they're coroutines (BCPL ones). Anything that blocks in exec (i.e. Wait()) will stop all of them. Internally, it switches around between them as needed (no preemption). CallCo() will swap to a coroutine (much like jsr), WaitCo() returns a results to the parent (much like rts). One important difference with subroutines is that the coroutines maintain their own stacks and state. ResumeCo() is basically a branch to a coroutine (as if it had been CallCo'd by your caller - much like exiting a subroutine with jmp some_other_subrutine). There are various other utility functions, like StartCo, KillCo, CreateCo, etc.

Each open filehandle is a coroutine. The master coroutine CallCo()'s it when it gets a packet or when IO comes back. Note that this is where the stack-based state information of the coroutine come into play - a filehandle coroutine does IO by queuing a request, and calling WaitCo(). The disk read/write coroutine pulls things from the queue and gets woken up when IO is done. When an IO is complete and verified, it notes which coroutine is waiting for the IO to complete (can be a list), and ResumeCo()'s that coroutine. Since the coroutine has a stack, it

```
continues at the point after the WaitCo with whatever it was
doing, now knowing the IO was complete. Note that the master
coroutine doesn't CallCo() an active filehandle coroutine, it
queues the packet for later handling.

If this seems simple, I guarantee you, IT'S NOT. It's mind-warping
and arcane when you need to care about the coroutines (but easy
and straightforward when you don't).  A lot of state information
becomes automatic, without having to build a (massive) explicit
state machine. I burnt out a lot of brain cells figuring out how
to handle locking of updates to hash chains in this setup, since
this required major interactions between coroutines. The end
result was simple, but it was a bitch to figure out.
```

1.13 version

```
5.7     AmigaDOS version command.

In order to set up a version number that the "version" command can
find, you need to declare a string starting with "$VER:". For
example:

char    version[] = "$VER: myprogram 1.1 (" __DATE__ ")";
```

1.14 path

```
5.8     Determining program path.
        [Doug Keller]

Here is code that will get the program's home directory even if
the program is resident.

struct Path {
    BPTR p_Next;
    BPTR p_Lock;
};
/*
**  getProgDir- get progdir: even if resident
**
*/
BPTR getProgDir(UBYTE *buffer, struct Library *DOSBase)
{
  struct Library *SysBase=*((struct Library **)4L);
  struct Path *path;
  struct Process *proc=(struct Process *)FindTask(NULL);
  struct CommandLineInterface *cli;
  BPTR lock, olddir, clock;

    if( proc->pr_HomeDir )
        {
            return( DupLock(proc->pr_HomeDir) );
        }
```

```
        if( cli = (struct CommandLineInterface *)BADDR(proc->pr_CLI) )
            {
            /* check if full path or current dir */
                SPrintf(SysBase,buffer, "%b", cli->cli_CommandName);
                if( lock = Lock(buffer,SHARED_LOCK) )
                    {
                        clock=ParentDir(lock);
                        UnLock(lock);
                        return( clock );
                    }

            /* check path */
                olddir = CurrentDir(NULL);
                for(path = (struct Path *) BADDR(cli->cli_CommandDir);
                path;
                path = (struct Path *) BADDR(path->p_Next) )
                    {
                        CurrentDir( path->p_Lock );
                        if( lock = Lock(buffer,SHARED_LOCK) )
                            {
                                UnLock(lock);
                                CurrentDir( olddir );
                                return( DupLock(path->p_Lock) );
                            }
                    }
                CurrentDir( olddir );

                /* check c: */
                if( clock = Lock("C:",SHARED_LOCK) )
                    {
                        olddir = CurrentDir(clock);

                        if( lock = Lock(buffer,SHARED_LOCK) )
                            {
                                UnLock(lock);
                                CurrentDir( olddir );
                                return( clock );
                            }
                        CurrentDir( olddir );
                        UnLock(clock);
                    }
            }
        return( NULL );
    }
```

1.15 exec

```
*** 6   EXEC INFORMATION
    6.1     What do GURU numbers mean?
    6.2     How do you debug an input handler / task?
```

1.16 guru

```
    6.1     What do GURU numbers mean?

    Get Stegan Zeiger's program "alert" from amiga.physik.unizh.ch or
    Fish disk #636. It will translate the 32-bit GURU values into
    something more understandable.
```

1.17 inputhandler

```
    6.2     How do you debug an input handler / task?

    a.  Create a message port in the parent process to which you can
        send messages from the handler.

    b.  Use the kprintf() routines in debug.lib, optionally trapping
        the output to a console with Sushi.
```

1.18 graphics

```
*** 7   GRAPHICS INFORMATION
    7.1     What is DIG/RTG?
    7.2     What is the format of IFF ILBM and ANIM files?
    7.3     Fast conversion between chunky and planar pixels.
```

1.19 dig_rtg

```
    7.1     What is DIG/RTG?

    DIG is Device Independent Graphics. A set of graphics routines
    that does not require a particular device to work.

    RTG is ReTargetable Graphics. A set of graphics routines that
    normally work with a particular device, but can be changed at a
    low level to work with other devices.

    They are basically the same thing, though DIG is more often used
    to refer to things like Postscript where the output device can be
    a monitor, a plotter, a printer, etc. RTG is more often used to
    refer to different display hardware, usually meaning a different
    number of colors or a different pixel format, planar vs chunky.
```

1.20 iff

7.2 What is the format of IFF ILBM and ANIM files?

The original format specs for these files can be found on Fish disk #185. That disk also includes public domain code for the creation of OPT 5 DLTA chunks for ANIM files.

Updated specifications for some of the formats, new additions since 1985, and example code are also distributed as the "NewIFF" archive, available on Fish disk #705.

It may not be necessary to know the exact format if you use a library such as iff.library for loading and saving pictures. This library also contains a routine for decoding DLTA chunks from ANIM files, including OPT J chunks from the old Sculpt 3D movies. OPT J chunks used to be proprietary.

1.21 chunky2planar

7.3 Fast conversion between chunky and planar pixels.

Some pointers to source are:

a. rot2.lha (on amiga.physik), by Jason Freund, Gabe Dalbec, and Chris Hames.

b. tmapdemo.lha (on amiga.physik), by Chris Green.

c. The Great Chunky to Planar Competition, run by James McCoull. Email jmccoull@postoffice.utas.edu.au for more details.

1.22 intuition

```
*** 8   INTUITION INFORMATION
    8.1     How do I block a window from user input?
    8.2     Does WindowToBack() interfere with Workbench icons?
    8.3     Can a program move the mouse pointer?
    8.4     Gadgets don't 'deghost' after OffGadget()/OnGadget()?!?
    8.5     Can I have gadgets on a Screen without a Window?
    8.6     What are all the different fonts in Intuition for?
    8.7     How do I get that snazzy 3D look for my screens under 2.0?
```

1.23 input

8.1 How do I block a window from user input?
[Peter Cherna]

Normally this is required when you want to open a second window that should act like a requester. The solution is to open a real 0x0 requester in the window you want to block. For example:

```
    struct Requester req;
    InitRequester( &req );
    Request( &req, window );
    ...
    EndRequest( &req, window );
```

1.24 windowtoback

```
8.2     Does WindowToBack() interfere with Workbench icons?
        [Markus Juhani Aalto / Peter Cherna]
```

There is a dysfunction between the user dragging icons and some Intuition functions such as WindowToBack, which affects programs that use these functions over a regular interval. For example, calling WindowToBack() once every four seconds can tickle this problem. Under KS2.0 and above, Workbench can overcome this. For previous versions of the OS, you have to try and lock layers before using WindowToBack().

```
    /* Lock layers. If someone else has locked it, then we wait. */
    LockLayerRom( WorkbenchScreen->LayerInfo.top_layer );
    Forbid();
    UnlockLayerRom( WorkbenchScreen->LayerInfo.top_layer );

    WindowToBack(MyWindow);     /* Or WindowToFront() */
    Permit();
```

Remember this is only useful if you are frequently calling these functions.

1.25 mouse

```
8.3     Can a program move the mouse pointer?
        [Jeff Dickson]
```

The mouse pointer may be moved under software control by the IECLASS_POINTERPOS input event. This input event is talked of in the RKM chapter on the Input Device. For a working example, see the article, "No Mousing Around" in the September 1992 issue of Amazing Amiga Technical magazine.

1.26 ghostedgadgets

```
8.4     Gadgets don't "deghost" after OffGadget()/OnGadget()?!?
        [Peter Cherna]
```

OffGadget() is very simplistic, and only works with certain kinds of gadget imagery, as it turns out. Basically, you want to write your own routine, whose outline is:

```
myOffGadget()
{
  RemoveGList( the gadgets to unghost )
  RectFill( the select boxes of those gadgets )
  for each gadget, clear GADGDISABLED
  AddGList( those gadgets )
  RefreshGList( those gadgets )
}
```

See the 2.04 RKM, p.128-130, for a detailed discussion of the issues involved in programmatic gadget refresh.

1.27 gadgetswowindow

8.5 Can I have gadgets on a Screen without a Window?

No. Normally what you do is open a borderless backdrop window on the screen and attach the gadgets to that.

1.28 fonts

8.6 What are all the different fonts in Intuition for?
[Peter Cherna]

	What you tell OpenScreen	Screen's Font	Windows' RPort's Font
A.	NewScreen.Font = myfont	myfont	myfont
B.	NewScreen.Font = NULL	GfxBase->DefaultFont	GfxBase->DefaultFont
C.	{SA_Font, myfont}	myfont	myfont
D.	{SA_SysFont, 0}	GfxBase->DefaultFont	GfxBase->DefaultFont
E.	{SA_SysFont, 1}	Font Prefs Screen text	GfxBase->DefaultFont

A and B are the options that existed in releases prior to Release 2.0.

C and D are new Release 2.0 tags that are equivalent to A and B respectively.

E is a NEW option for Release 2.0. The Workbench screen uses this option.

GfxBase->DefaultFont will always be monospace. This is the "System Default Text" from Font Preferences.

The "Screen Text" choice from Font Preferences can be monospace or proportional.

'myfont' can be any font of the programmer's choosing, including a proportional one. This is true under all releases of the OS.

The menu bar, window titles, menu-items, and the contents of a string gadget use the screen's font. The font used for menu items can be overridden in the item's IntuiText structure. Under Release 2.0 and higher, the font used in a string gadget can be overridden through the

```
StringExtend structure.  The font of the menu bar and window titles
cannot be overridden.

The screen's font may not legally be changed after a screen is opened.

IntuiText rendered into a window (either through PrintIText() or as a
gadget's GadgetText) defaults to the Window RastPort font, but can be
overridden using its ITextFont field.  Text rendered with the Text()
graphics.library call appears in the Window RastPort font.

The Window's RPort's font shown above is the _initial_ font that
Intuition sets for you in your window's RastPort.  It is legal to
change that subsequently with SetFont().
```

1.29 3dlook

```
   8.7     How do I get that snazzy 3D look for my screens under 2.0?
           [Peter Cherna]

   You must give Intuition a ~0-terminated "pen-array" if you want
   the full 3D look on your screen.  The pen-array is used to inform
   Intuition and other interested software (including GadTools) what
   color pens should be used for different functions.  Here are the
   pens, and their value and color for the regular Workbench are
   shown in brackets.

DETAILPEN - Same as screen's detail pen.  (0, gray).
BLOCKPEN - Same as screen's block pen.  (1, black).
TEXTPEN - Text color, when rendered over background (1, black).
SHINEPEN - Bright 3D edge (2, white).
SHADOWPEN - Dark 3D edge (1, black).
FILLPEN - Used to fill highlighted or selected areas (3, blue).
FILLTEXTPEN - Text color, when rendered over FILLPEN (1, black).
BACKGROUNDPEN - Must be zero currently. (0, gray).
HIGHLIGHTTEXTPEN - Not used much, special text over background. (2, white).

UWORD myPens[] =
{
   0,  /* DETAILPEN */
   1,  /* BLOCKPEN */
   1,  /* TEXTPEN */
   2,  /* SHINEPEN */
   1,  /* SHADOWPEN */
   3,  /* FILLPEN */
   1,  /* FILLTEXTPEN */
   0,  /* BACKGROUNDPEN */
   2,  /* HIGHLIGHTTEXTPEN */
   ~0, /* terminator */
};

...

   myScreen = OpenScreenTags( &myNewScreen,
       ...                     /* Insert your tags here */
       SA_Pens, myPens,
```

```
        ...                     /* and/or here */
        TAG_DONE );
```

Or, if you want to use OpenScreen(), to remain 1.3-compatible, you can do the following:

```
struct TagList mytags[] =
{
   SA_Pens, 0,
   TAG_DONE, 0,
};

struct ExtNewScreen myExtNewScreen =
{
 /* initialize as usual */
};

   mytags[0].ti_Data = myPens;

   /* V1.3 ignores the Extension field, but 2.0 notices it
    * if NS_EXTENDED is set.
    */
   myExtNewScreen.Extension = mytags;
   myExtNewScreen.Type |= NS_EXTENDED;

   myScreen = OpenScreen( &myExtNewScreen );
```

If you will be using the same color values as Workbench, you can also use its pens, by defining:

```
UWORD myPens[] =
{
   ~0, /* terminator */
};
```

This pen-array has no data, just a terminator. Intuition will use Workbench's pens for any pens you don't specify because you terminate your array early. If you do this, you'd better be using the Workbench colors, though.

1.30 sas

*** 9 SAS C PROBLEMS

9.1 How do I get rid of the CON: created from Workbench?

The window is opened by the ___main() function. Refer to Appendix 1 of the SAS C User's Guide, Volume 1.