ixemul.info

COLLABORATORS			
	TITLE : ixemul.info		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 11, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ixemul.info

1.1 IXemul.library

```
                    IXemul.library
                          **************


     Introduction       Why IXemul.library?
     Installation       How to Install it
       Copyright        Copyright Restrictions
      GNU License       The GNU General Public License
        Authors         The Authors of IXemul.library
     Configuring        Configuring IXemul.library to your tastes
       Tracing          Tracing Your Programs with IXemul.trace
     Recompiling        Recompiling IXemul.library
   Porting UNIX Apps    Porting UNIX Applications
   IXemul's Signals     Implementation of BSD Signals
         Bugs           Where to Report Bugs or Suggestions
         FAQ's          Frequently Asked Questions
```

1.2 IXemul.library/Introduction

```
Why IXemul.library?
*******************

  IXemul.library was originally written to provide an environment
for the porting and subsequent compilation of UNIX C programs. The design
of the library was therefore guided towards UNIX/BSD compatibility, and *not*:

 o To be too conservative with resources.

 o To be particularly conformant to Amiga habits. Thus if I had to decide
   whether I should make a function act more like an Amiga function or
   more like a UNIX/BSD one, I decided for the latter. As an example:
   _cli_parse() does wildcard expansion, and tries to apply more or less
   UNIX/BSD shell semantics to an argument line, it doesn't call
   ReadArgs().
```

```
   The types used in my own source code are all from sys/types.h (except
   BPTR). I don't think capitalized identifiers should be used for typedef'd
   types. According to C-conventions, anything written in captials should
   be '#undef'inable, which typedefs aren't. Thus if you write contributions
   to be included into the official distribution of this library, code
   according to this. Use 'u_char' and not UBYTE, etc. I don't care that
   this is against the Commodore coding standard, this is my code, and
   I decide what I like and what not.

 o To be particularly suited for inclusion into a shared library, although
   most things *are* shared now. What I'd really want for the Amiga is
   the concept of a dynamic linker.

On the other hand, it is:

 o Expandable. As an example, a file descriptor already can refer to 'real'
   files, directories, memory buffers treated as files.

 o Patchable. If you want some function to behave differently, you can
   SetFunction() it, and the rest of the library should use your new entry.
```

1.3 IXemul.library/Installing IXemul.library

```
Installing IXemul.library on your system
****************************************

  In the libs directory of this release are 4 versions of
IXemul.library:
    1) IXemul.library.000
    2) IXemul.library.020noffp
    3) IXemul.library.020ffp
    4) IXemul.library.030

  There is also a special version of IXemul.library called
ixemul.trace.  More on this later.

  If you have a 68000 or 68010: Rename IXemul.library.000 as
IXemul.library and place it in your libs: directory.

  If you have a 68020 or a 68030 and NO math coprocessor: Rename
IXemul.library.020noffp as IXemul.library and place it in your libs:
directory.

  If you have a 68020 and a 68881 or 68882: Rename
IXemul.library.020ffp as IXemul.library and place it in your libs: directory.

  If you have a 68030 (and a 68881 or 68882) or a 68040: Rename
IXemul.library.030 as IXemul.library and place it in your libs: directory.

============================================================================

  Next, copy the files in lib to gcc:lib and the files in
lib/libb to gcc:lib/libb.
```

If you have a 68020 or higher, also copy the files in lib/lib020 to gcc:lib/lib020 and the files in lib/lib020/libb to gcc:lib/lib020/libb.

Copy the include files to their respective places in gcc:include.

IXemul.trace provides a means of tracing the library calls made by your program and printing them in a CLI window. Just copy this file into your libs: directory for now. See IXemul.trace for more details.

1.4 IXemul.library/Copyright

Copyright Restrictions
****

1.5 IXemul.library/Authors

```
Authors
*******

  Some of the code in the library is taken from NetBSD, written
by various authors.  These files separately are Copyright (C) by The
Regents of the University of California.  Their inclusion in the
library is governed by their restrictions.  See   Copyright  for the
restrictions.

  Also, some of the code in the library is taken from other
software under the GNU General Public License and is thus governed
by their restrictions as well.  See   Copyright  for the
restrictions.

  The original version of ixemul.library was written by Markus
Wild.  He now presently works on NetBSD, a UNIX port for the Amiga.
For those interested in running a Public Domain UNIX, you can get
the necessary files ftp from: ftp.iastate.edu or from
sun-lamp.cs.berkeley.edu.

  This version was built and updated by Rafael Luebbert. He
can be reached VIA E-mail at LuebbeRW@lp.musc.edu.

  The following people have contributed files to be included in
the IXemul.library:

  Leonard Norgaard
  Ray Burr

  Also, thanks to all beta-testers and everyone who reported
bugs in previous releases.
```

1.6 IXemul.library/Configuring IXemul.library

```
Configuring IXemul.library to your tastes
*****************************************

Ixconfig is used to tailor the library to your requirements and/or
habits. Just running ixconfig without options prints the current settings,
which look like this by default:

    1> ixconfig
    Translate . and .., translate /, don't translate symlinks,
    allow AmigaDOS notation, membuf size = 0,
    red zone size = 0, stack watcher is disabled (and not active).

Here's an explanation of those settings:
"translate . and .."  mapping of `a/./b/../c' into `a/b//c' is enabled
"translate /"   mapping of `a///b' into `a/b' and `/device' into
      `device:' is enabled. Note: You can't currently get
      a directory of the virtual `/' directory this way.
```

```
"translate symlinks"  apply 'translate /' to contents of symlinks as well
"AmigaDOS notation" allow use of device names in the colon form
      (ie. sys: instead of /sys), and don't force '..'
      notation.
"membuf size"   if you set a non-zero value here, all files upto
      that value, that are opened O_RDONLY are read
      into memory, and read/seek operations occur in memory.
"red zone size N" size of 'safety net'. If your program uses
      so much stack, that the stack pointer is more
      than N bytes near the stack bottom, your program
      is sent a SIGSEGV signal. Red zone size is used
      when starting a new process, if you change it later,
      no already running processes are affected.
"stack watcher"   global toggle. If disabled, no SIGSEGV signal is
      sent to any program (but if red zone size is > 0,
      the process keeps a pointer, so that if you reenable
      the stack watcher, SIGSEGV will be sent again).

This was an explanation of the output of ixconfig, to change those values
type 'ixconfig -h' for an explanation on the available switches. One
switch might need further explanation: '-s'. If you specify '-s', ixconfig
goes to sleep after setting the new parameters, and won't return until you
break it with ^C. This is the preferred switch if you run ixconfig from your
startup-sequence in the background, as then your changes can't be undone by
flushing the library (ixconfig keeps it open, so that Expunge() can't flush
it).
```

1.7 IXemul.library/GNU License

```
      GNU LIBRARY GENERAL PUBLIC LICENSE
           Version 2, June 1991

 Copyright (C) 1991 Free Software Foundation, Inc.
                    675 Mass Ave, Cambridge, MA 02139, USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the library GPL.  It is
 numbered 2 because it goes with version 2 of the ordinary GPL.]

          Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Library General Public License, applies to some
specially designated Free Software Foundation software, and to any
other libraries whose authors decide to use it.  You can use it for
your libraries, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
```

this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the

users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may

distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply,

and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the library's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This library is free software; you can redistribute it and/or
> modify it under the terms of the GNU Library General Public
> License as published by the Free Software Foundation; either
> version 2 of the License, or (at your option) any later version.
>
> This library is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
> Library General Public License for more details.
>
> You should have received a copy of the GNU Library General Public
> License along with this library; if not, write to the Free
> Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the
> library 'Frob' (a library for tweaking knobs) written by James Random Hacker.
>
> <signature of Ty Coon>, 1 April 1990
> Ty Coon, President of Vice

That's all there is to it!

1.8 IXemul.library/IXemul.trace

```
Tracing Your Program Using IXemul.trace
***************************************

This is an attempt at a tracer for ixemul.library. Since it scans every
call made thru the library base, it catches even more calls than for
example SnoopDOS.

Since tracing support slows down *all* clients of the library (as each
call is now routed thru tracing functions), there are two library
versions. ixemul.library itself doesn't contain tracing support, using
the tracer with this library gives you a "Function not implemented"
error. To use the tracer you have to load ixemul.trace as the main
library. See below for more detailed instructions.


The tracer itself is quite straight forward, a lot more calls could use
more descriptive parameters, its mostly typing work, but I didn't feel
like providing more ioctl, open, pipe like print functions ;-))

The option you're most certainly going to use is '-m', as the library
uses sigsetmask() internally a lot, and its mentioning would just clutter
the overwhole display.


Short instructions to get the tracer up
---------------------------------------
o Flush ixemul.library out of the system.  I use "avail flush".  If you
  are using ixconfig to keep the library loaded so your defaults don't
  get changed, you'll have to reboot.

o Make sure the system loads ixemul.trace instead of ixemul.library
  next time it tries to fire up ixemul.library. I usually do this by
  renaming IXemul.library to IXemul.lib, and then renaming IXemul.trace
  to ixemul.library.  DON'T FORGET TO SWITCH BACK LATER!

o Start the tracer (in bin:trace).  The following options are currently
  recognized (see bin:trace.c as well):

    -a  print all calls, even those considered 'not interesting', for
        example internal stdio calls, 32bit arithmetic emulation for 68000,
        and such stuff you generally don't want to see.
    -i  normally, the tracer shows a function call when that function returns,
        to be able to display the return value (and 'errno' in parenthesis).
        When you specify '-i', it always displays function on entry, instead
        of on exit. See below for current problems without the -i option!
    -m  skip sigsetmask calls. This function is used extensively inside
        the library, and will in most cases just clutter the tracer output
        with information you don't want to see.
    -o  specify a logfile. If -o is omitted, output goes to stdout.
    -p  only print output from a specific process. You have to provide the
        decimal address of the process (ok ok, this is not too user-friendly,
```

```
        I'm open to better alternatives ;-)). Default is to scan all current
        ixemul clients.
    -s  only print calls to specific system call. You have to specify the
        number as found in <sys/syscalls.def>. Again, this option could be
        much more user-friendly, but I'm a lazy guy;-)

You should now see a log of all functions involved from your program(s),
and some functions involved as a consequence of your functions from inside
the library. The output means:

$7588790: <strcmp("foo", "-n")=57 (0)
  ^       ^       ^            ^   ^
  |       |       |            | errno value
  |       |       |          result of function
  |       |     parameters
  |     type of log, < means exit, > means entry
process address


Some known problems
-------------------
There seems to be some race condition, that can result in deadlock when
running without the -i flag. I don't know, where the problem lies, such
stuff is really hard to find...

Currently, if trace-on-exit is enabled, the function is called with
16 arguments, no matter how many arguments you passed to it. This should
be enough for 99% of all applications, but you may have more than 16 arguments
to a printf() somewhere.

Functions returning more than a 4byte result *don't* work unless you specify
the -i option!!
```

1.9 IXemul.library/Recompiling IXemul.library

```
Recompiling IXemul.library
**************************

  First, you'll need RCS, or a port of it.  I use HWGRCS which
is wonderful.  You'll also need to get bsd-ld, which supports reloc-sizes
that ld does not.

  Sources are written for compilation by GCC. You might be able to
compile 95% of them with any ordinary ANSI-C compiler, but there are
cases where you have to change things for non-GCC environments
(mostly asm() situations).

  To regenerate the library, cd to ixem40.3/ and type allclean.
This is a batch file that will clean any older version objects you
have in the subdirectories.  Then choose the allmake file appropriate
to your hardware and type "allmake" then go make a sandwich and watch
some television - it will take some time.

  For 68000 users, you'll have to find GNU's gas 1.36 as gas 2.3
incorrectly parses aligns in some assembly.  You will also have to
```

```
recompile some files and assemble them with gas 1.36.  See the file
"readme.68000.recompile" for more details on which files.

  To regenerate libc.a, cd into ixem40.3/ and type 'libcmake'.
Wait for it to finish (can take quite some time, it has to create more
than 400 files !).

As an explanation to the separate compilation of crt0.o:  this has to be
compiled with '-fwritable-strings', since you have to be sure that ENTRY()
is the first thing in the generated object file.  If you don't specify
'-fwritable-strings', you'll get string constants at the first executable
address in your programs, and this will get you (and your computer) into
meditation if you try to execute such programs ;-))
```

1.10 IXemul.library/Signals in IXemul.library

```
How BSD signals are implemented
*******************************

I tried to implement as much of Berkeley style signals as possible on the
Amiga. This includes a trap handler as well as an asynchronous signal
facility. The one thing not implemented are interruptable system calls.
Since there are no 'real' system calls on the Amiga (ie. no calls that are
executed in Supervisor mode), those calls cannot normally be interrupted,
ie. forced to return to their caller. So all functions except sigpause()/
sigsuspend() will return to where they were interrupted if a signal
occurs.

These 32 new signals are 32 really new signals, not tied to any of the 32
Amiga signals provided by Exec.  The one exception is SIGBREAKB_CTRL_C,
which is by default bound to generate a SIGINT.

Signal handlers are called with the following arguments:
  void
  signal_handler (int signo, int code, void *address, struct sigcontext *sc)

Where
  signo:  is the signal number that occured, see <signal.h>
  code:   is a more specific characterization of signo available with some
    signals. It is available with all signals that are generated
    because of a processor exception, and then contains the format
    identifier of the exception frame (this is correct even for the
    68000, where such an identifier is faked, ie. it doesn't really
    exist). Thus a 'division by zero' exception would be invoked by
      signal_handler (SIGFPE, 0x2014, address, sc)
  address:address refers to the instruction that caused the signal.
  sc:     please don't use sc, as it may change in the future. It contains
    the context to restore after the signal handler returns.

*****************************************************************************
If you use signals in your own code, make sure that you never allow a
situation, where when your program is interrupted resources stay allocated!
*****************************************************************************
```

That is, the following example is BAD :

```
  ..
  fh = Open ("foobar", MODE_OLDFILE);
  if (fh)
    {
      .. do something with it ..
      Close (fh);
    }
```

If your program is interrupted and terminated after you got your file handle, ‘fh’ will never be closed! There are two solutions to get around this problem, either use library functions from ixemul.library, or explicitly mask signals while you have resources locked. Thus in this example, either do:

```
  fd = open ("foobar", O_RDWR);
  if (fd >= 0)
    {
      .. do something with it ..
      close (fd);
    }
```

in that case the library will do resource tracking on fd. Or explicitly mask the signals:

```
  omask = sigsetmask (~0);  /* mask all signals */
  fh = Open ("foobar", MODE_OLDFILE);
  if (fh)
    {
      .. do something with it ..
      Close (fh);
    }
  sigsetmask (omask);   /* reset the mask */
```

Note that the second solution is worse than the first one, because the user may send the process a non-maskable signal that would terminate the process unconditionally (SIGKILL does this), and don’t forget that the user isn’t able to break your program as long as you have signals masked!

Ixemul.library does resource tracking on all file-related functions (create(), open(), dup(), pipe()) and on memory allocations thru malloc() and realloc(). Thus if you use those functions instead of dos.library and exec.library functions, you don’t need any clever resource tracking stuff to do on your own, that’s what the library is for ;-)

If you use Amiga specific resources like Windows and Screens from Intuition, make sure to add an atexit() handler to close those resources, if the user should decide to interrupt your program. Before the program is left, the chain of registered atexit-handlers is called in exit(). So PLEASE NEVER EVER call _exit() if you have registered any custom atexit() handlers. It is a bad habit anyway, but normally you may call _exit() without resource loss (stdio won’t flush its buffers, but that’s about all), as long as you close ixemul.library after use, and this IS A MUST, as for every Amiga shared library anyway.

I provided a new unique Amiga specific signal called SIGMSG. If you set up a handler for this signal, then

 o the default mapping from SIGBREAKB_CTRL_C into SIGINT will no longer

```
    occur
 o  your handler is called with the following arguments
      signal_handler (SIGMSG, new_exec_signal_mask)

In this case, you have to deal with Exec signals yourself, so don't forget
to clear those signals that you want to receive notification about again
later.
Thus if you'd want to handle SIGBREAKB_CTRL_C yourself, don't forget to

  SetSignal (0, SIGBREAKF_CTRL_C)

at the end of the handler, or you'll never get notification about that
signal again.


If your program is interrupted by a signal and the default action of that
signal is to terminate your program, and you didn't set up a handler to deal
with that signal, your program is terminated by calling `exit (128 + signo)'.
There are no core-dumps yet, I first have to think about a useful format
for a debugger that takes care of the Amiga's memory architecture.


The signal implementation uses some of the Berkeley kernel sources of the
4.3BSD-reno release for the hp300. I didn't disable everything that isn't
implemented currently, so you might face strange behavior if you currently
try to send a SIGSTOP to a process using the library, you better not ;-))

Currently supported are the following signals:
  SIGINT:    bound to ^C (SIGBREAKB_CTRL_C) unless there is a SIGMSG handler
  SIGILL:    generated by some hardware exceptions
  SIGFPE:    generated by some hardware exceptions
  SIGBUS:    generated by some hardware exceptions
  SIGALRM:   if you use alarm() or the ITIMER_REAL interval timer
  SIGVTALRM: if you use the ITIMER_VIRTUAL interval timer
  SIGPROF:   if you use the ITIMER_PROF interval timer
  SIGMSG:    if you provide a signal handler for it
  SIGCHLD:   a vfork()'d child died (or stopped ?;-))
  SIGSEGV:   the programs used more stack than you allowed it to (see below)

more are to follow.  You may send any of the 32 signals to a process using
the library with the `kill ()' function, the default behavior of a process
is described in a UNIX/BSD man page for signals.  As mentioned above,
stopping a process isn't currently implemented, and may produce strange
behavior.
```

1.11 IXemul.library/Porting Unix Applications

```
Porting UNIX Applications
*************************

  Several functions are missing from IXemul.library, mainly those
that depend on UNIX's unique memory handling.  sbrk and brk are old
archaic functions that should be replaced in any new ports of UNIX
applications anyway.
```

UNIX/BSD process management is one of the most nasty design differences between AmigaDOS and UNIX/BSD. 'fork()' for example is hardly possible to implement on AmigaDOS, as it requires to create an identical copy of the parent process. This is only feasible with virtual memory, where processes can be mapped at equal places in memory. Under AmigaDOS this would have to be simulated by copying of stack and malloc'd data whenever a process is activated, and copying them to a safe place before it is disactivated.

This problem can be avoided, if the program to be run under AmigaDOS is only fork()ing, because it just wants to start another process. In that case, no such copying as described before is necessary, and BSD therefore invented the 'vfork()' function, which works like 'fork', but runs the child on the parents memory segments (stack and malloc'd data). While the child is using the parents resources, the parent is sleeping in a not interruptible state.

That much for theory;-) I tried to implement an as compatible as possible vfork() function, that behaves like the BSD one.

Since I won't try to implement 'fork', I provided a possible alternative (you tell me;-)). As an extension, you get the 'vfork_resume()' function, which causes the parent to resume, just like it would if you called '_exit()' or one of the 'exec*()' functions. Since this function is quite dangerous (and an even bigger hack than vfork() itself..), here's what's happening in 'vfork_resume()':

- the child switches to its own stack. After vfork(), the child is using the stack of the parent process. Since no two processes can share the same stack in parallel, vfork_resume() causes a switch to the 'real' stack of the child.
- the parent is sent a wakeup message.
- both processes run concurrently

The first point is the most important one: Since vfork_resume() changes the stack pointer of the running process, you can't refer to any variables or parameters anymore after calling vfork_resume()! Only register variables survive such a call, and you have to explicitly store values in register variables that are subject to survive!

There's another potential problem with vfork_resume():

****
Don't exit() from the parent before all vfork()'d children have died!!
****

Since exiting from the parent causes the parents code and data segments to be deallocated, the child would find itself without code space to run on, and would probably cause a severe machine crash!

So always call at least 'wait (0)' before returning from the parent.

exec*()
=======

In most cases, you just use 'vfork()' to later overlay the process with
a new image, that is you want to start another program. The way AmigaDOS
loads processes is not too well suited to do 'exec' style program starting,
yet it is possible, although with slight resource wasting..

First problem is, that all exec* functions pass an argument vector to
the new program, whereas AmigaDOS programs expect to be passed an argument
line (instead of the vector of arguments). Since in my opinion it would
be a good thing if a program could get an argument vector directly (in that
case the inherent problem of passing multi word arguments to a program would
be finally solved, no more weird quoting needed!). That's why I provided
a mechanism that allows this vector passing, and it works like this (look
at crt0.c for a concrete implementation of this concept!):

  The program has to provide a magic header at the first executable location
  in its code. This magic header looks like this:
    - JMP instruction to common AmigaDOS startup
    - struct exec area. Use the OMAGIC a_magic code.
    - provide an alternate entry vector in a_entry. execve() jumps thru
      this vector to pass vectors to your program, instead of going
      thru the normal AmigaDOS startup part.

As long as you use my crt0.o and libc.a, this whole thing is completely
transparent to your program. You only have to care for it if you want
to support the mechanism in other languages as well.

The second problem is how to start 'old' AmigaDOS programs from execve().
If the program has the described magic header, starting is easy. Else
another approach is taken. Since the new program can't refer to the
real file descriptors (I can't pass the open library without my startup
code), I have to setup DOS fields to use my filehandles. This may succeed
or not, depending on whether the descriptors in question are realized by
DOS files or not (in the future a not-compatible alternative would be
descriptors that refer to sockets!). Actual starting of the program is
done with RunCommand().

1.12 IXemul.library/Where to Send Bug Reports and Suggestions

Where to Send Bug Reports and/or Improvements in Code
****

  If discover any bugs in ixemul.library, please send me e-mail
with the following information:

  1) Your hardware configuration.
  2) The version of ixemul.library you are using.
  3) The function called when the crash occurred (using ixemul.trace)

  If you have any suggestions or improvement in the code, please
send them to me so they can be included in the next version.

  My E-Mail address is: LuebbeRW@lp.musc.edu
  I also read posts made to amiga-gcc-port@lists.funet.fi

1.13 IXemul.library/Frequently Asked Questions

```
Frequently Asked Questions
**************************

1) How can I inline IXemul.library's functions like the Amiga's functions?

  You can't.  The library is designed to be used by C, and not by
assembly.  So parameters are passed on the stack rather than in
registers.  This also means that there is no 'fd' file, and you can't use
any current library call pragmas to access its functions.  Recall though,
that calling functions of ixemul.library inline will not result in an
order of improvement as calling standard library functions inline.  The
glue functions don't have to shuffle arguments from and to the stack,
they just do a jump over the base table and are therefore very short
and very fast.

2) Can't you take out that damn trap handler so I can trace my programs?

  I plan on releasing a patch to disable the trap handler.  In the
meantime, you can use ixemul.trace, and you can use PowerVisor 1.42 with
the dirty mode enabled.
```