PhxLnk

COLLABORATORS

	TITLE : PhxLnk	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 22, 2024	

REVISION HISTORY

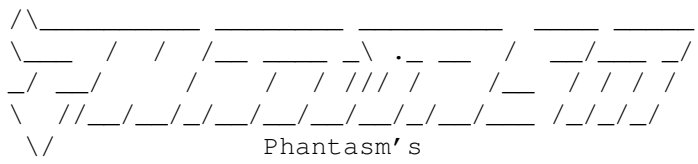
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PhxLnk

1.1 PhxLnk V4.31 (23-Sep-97)



P h x L n k V4.31

AMIGA-DOS Module Linker

Contents

1.2 Introduction

PhxLnk was written in pure assembler-code, assembled with PhxAss and linked with BLink (first version) and PhxLnk itself. It supports all features of a standard Amiga-DOS linker (like BLink), except overlay hunks.

1.3 History since V3.00

V4.31 (23-Sep-97)
Because of an unaligned structure, the last two PhxLnk versions crashed on 68000 and 68010.
HUNK_DEBUG blocks are accepted before a code, data or bss hunk, but are ignored then.

V4.30 (02-Feb-97)
DEFINE/K supports assignment of absolute and relocatible XDEF symbols.
By setting the new switch ALV/S, PhxLnk is able to resolve 16-bit references between different sections by generating Automatic Link Vectors (ALV).
Additionally, the following arguments are recognized for BLink/SLink-compatibility, but have no effect: BATCH/S, ADDSYM/S, NOALVS/S, NOICONS/S, LIB/S.
New linker symbol _SMALL_DATA_LEN_ reflects the total small data size in bytes, when in small data mode and is zero otherwise. So it can also be used as an indicator for activated small data mode.

V4.26 (09-Jan-97)
Two new options, NOSHORTRELOCS/S and DONTSHORTENSECT/S for separately disabling RELOC32SHORT hunk blocks and the deletion of zero bytes, at the end of each section.
New linker symbol _CODE_LEN_ reflects the size of the first code section in bytes.

V4.25 (08-Nov-96)
The search for XDEF symbols was enhanced by use of a large hash table. The hash table's size can be modified with the HT=HASHTAB/K/N option.
Ctrl-C now works everywhere and not only during displaying the list of unknown symbols.
The EXT_REF types EXT_ABSREF8 ($8b) and EXT_ABSREF16 ($8a), which were introduced by OS3.0, are supported.
Norwegian catalog.

V4.24 (06-Jul-96)
Multiple defined XDEFs in object files are recognized and displayed.

V4.23 (03-May-96)
LoadSeg() V40 doesn't like a section length of zero, because it will refuse to clear the bss-part of a section in this special case. Now, a minimal section length of one longword is guaranteed.
PhxLnk uses its own buffered I/O routines instead those which the dos.library offers. The size of the buffer is adjustable by the new CLI parameter BUFSIZE/K/N, which defaults to 8192.
Creating an FFS file is more than five times faster as with v4.22.

V4.22 (23-Mar-96)
The maximum length of the destination file name was much too small and was redimensioned to 233.
The XREF sub type EXT_RELREF32 ($88) for 32-Bit PC-relative references is supported.
PhxLnk uses buffered I/O (FWrite) for creating the destination file.

V4.21 (08-Mar-96) PhxLnk couldn't handle multiple dots in a file name. So the object file "a.b.c.o" became "a" instead of "a.b.c".
The listing of undefined symbols may be interrupted with CTRL-C at any time.

V4.20 (20-Dec-95) Since v4.20 there are two versions of PhxLnk. One for OS2.x and the other for OS3.x. This splitting is a result of the intensive usage of MemPool functions, which are part of the exec.library under OS3.x (the OS2.x version gets their MemPool functions from amiga.lib). The MemPool adaption was done by Volker Barthelmann <volker@vb.franken.de>. It enables library-linking with a multiple speed.
There were some crashes, when empty sections, which contained some required XDEFs, were automatically removed. The only possibility to prevent it was by setting the PRESERVE/S switch.

V4.17 (24-May-95) Partial support for the SLink (SAS/C). PhxLnk defines ___ctors and ___dtors as NULL to avoid errors during linking. Complete SLink-support may follow in a later release. (if somebody needs it :-)

V4.16 (18-May-95) Fixing the bug with __MERGED sections in V4.10 generated a new one in the 'normal' small data mode.
The shortening of sections, as introduced in V4.00, doesn't work very well.

V4.15 (19-Mar-95) PhxLnk can generate RELOC32SHORT blocks.

V4.10 (21-Feb-95) Linking of __MERGED sections had a bug.

V4.03 (09-Feb-95) When PhxLnk discovers a read error, while reading a '@'-file, it crashed.

V4.02 (25-Jan-95) PhxLnk can read the object and library names from one or several files.

V4.00 (18-Nov-94) PhxLnk V4.00 requires OS2.04 as a minimum. As a result it offers the standard ReadArgs()-Command Line parsing and is much shorter.
DEBUG Hunk blocks are treated the same way as with SAS/C's BLink, which gives the possibility to generate load files for a Source Level Debugger.
The latest version of the PowerVisor Debugger, V1.42, unfortunately has still some problems with BLinked programs, consisting of several source files, so there is a compatibility switch, called PVCOMPAT.
The new argument DEFINE (see CLI Parameters) gives the possibility to define an absolute Linker-

Symbol (quite similar to the small data symbols supplied by PhxLnk).
By utilitzing the new switch BLINKCOMPAT PhxLnk will treat small data modules the same way like SAS/C's BLink does.
Data and Bss sections, which were named "__MERGED", will be coalesced into a small data section (without having to specify the SMALLDATA switch).
Zero bytes at the end of a Code or Data section will be ignored, which shortens the resulting load file. Because that doesn't work under Kickstart 1.x, there is a compatibility switch called KICK1.

V3.10 (04-Aug-94) Fixed a bug with catastrophic proportions, which sometimes appeared when linking with libraries.
To be honest: I don't think, that one of the pre-V3.10 versions are safe enough to link libraries ;-)
PhxLnk was completely localized. Until now, german and polish catalogs are available.
Documentation was converted into Amiga-Guide format.

V3.05 (31-Jul-94) Fixed a linker-library bug: Sometimes sections of a library, though not included, appeared in HUNK_HEADER with random length.
HUNK_RELOC and HUNK_SYMBOL of zero length will no longer be included.

V3.01 (22-Jan-94) Because of massive changes in V3.00, there was a little bug with the name of the output file.

V3.00 (18-Jan-94) Fixed many problems with library linking, which could lead to a FreeMemoryTwice Guru (or even worse).
Some linker symbols of Lattice/SAS (_LinkerDB, __BSSBAS, __BSSLEN) and DICE (__RESIDENT,__DATA_BAS, __DATA_LEN,__BSS_LEN) are supported.
The special library format of Lattice/SAS, using HUNK_LIB and HUNK_INDEX, is also supported. PhxLnk converts them into standard library format.

1.4 Instructions

1.5 Sytem Requirements

Since PhxLnk V4.00 you *must* have OS2.04 (V37) as a minimum. This makes PhxLnk much shorter, faster and easier for me to code. I don't think that

this limitation (which is a progress in my eyes) doesn't hurt anybody nowadays.

1.6 Starting PhxLnk

Normally, PhxLnk is started from your shell. You should copy PhxLnk from the OS2.x directory, if you have OS2.x installed, or from the OS3.x directory otherwise, and copy it to C: or define a path or link. The OS2.x version will run on OS3.x Amigas too, but it's a bit larger.

```
Format:   PhxLnk [FROM] {<object module|library module>}
          [TO <output file>] [SMALLCODE] [SMALLDATA] [NODEBUG] [CHIP]
          [PRESERVE] [PVCOMPAT] [BLINKCOMPAT] [KICK1] [MAXSECTS=<n>]
          [BUFSIZE=<n>] [HASHTAB=<n>] [NOSHORTRELOCS] [DONTSHORTENSECT]
          [BATCH] [ADDSYM] [NOALVS] [NOICONS] [LIB] [ALV]
          [DEFINE "<symbol>[=value][,<symbol>...]"]

Template: FROM/M,TO/K,SC=SMALLCODE/S,SD=SMALLDATA/S,ND=NODEBUG/S,CHIP/S,
          PRESERVE/S,PV=PVCOMPAT/S,B=BLINKCOMPAT/S,K1=KICK1/S,
          MAXSECTS/K/N,BUFSIZE/K/N,HT=HASHTAB/K/N,NOSHORTRELOCS/S,
          DONTSHORTENSECT/S,BATCH/S,ADDSYM/S,NOALVS/S,NOICONS/S,LIB/S,
          ALV/S,DEF=DEFINE/K
```

Starting PhxLnk with no argument or with a single '?' will display a short description. For a more precise description, refer to Parameters.

There are three types of modules which can be linked:
- Object modules with extension ".o" or ".obj" which normally consist of one unit. PhxLnk also links object modules with several units.
- Library modules with extension ".lib" which can consist of any number of units. PhxLnk will only include units, if at least one ext_def-symbol is referenced in an object module unit or in an already included library unit.
- Lattice/SAS Extended Library modules (also with ".lib" extension). They are translated into the standard library format by PhxLnk (not a very good solution - but it works).

Names with another extension will be rejected.

The module names can appear in any order, provided the first is an object module which contains the startup code.

IMPORTANT!
Load files created by PhxLnk are NOT Kickstart 1.x compatible by default! PhxLnk deletes zero-bytes at the end of a section and tries to use the much shorter RELOC32SHORT blocks, unless you set the KICK1 switch.

1.7 Description of command line parameters

FROM/M — All parameters without a keyword specify the names of the object and library modules to link. For valid name extensions refer to Starting PhxLnk.

Names which start with an '@' specify the name of an ascii file, which contains object and library names (or even more '@'s, if you like). These names can be separated by blanks, tabs, linefeeds or whatever you want. '"' are supported.

TO/K Determines the name of the output file to be produced. If not specified, the output file has the name of the first module without its extension. Example:
"PhxLnk prog1.o prog2.o c.lib m.lib" will generate a load file with the name "prog1".

CHIP/S This switch forces all sections to be loaded to Chip memory.

PRESERVE/S The normal case is, that PhxLnk removes all section with zero length to save memory. Choose this switch, if you want to preserve empty sections.

B=BLINKCOMPAT/S PhxLnk will be compatible to BLink, when linking small data modules. That means, that if the small data section is smaller than 32k, PhxLnk will use a small data pointer (_LinkerDB) which points to the beginning of this section instead 32766 bytes into it. As a result all near-offsets will start with 0 instead -32766.

K1=KICK1/S PhxLnk creates a load file which is compatible to Kick-start 1.x. That will prevent PhxLnk from deleting zero-bytes at the end of a section or trying to generate these nice RELOC32SHORT blocks.
Has the same effect like setting both NOSHORTRELOCS/S and DONTSHORTENSECT/S.

SC=SMALLCODE/S Normally only the sections with the same type and name will be coalesced. This switch makes PhxLnk to ignore the names of Code sections and to produce one large Code sections. Usually SMALLCODE is chosen, when using the small code model with your assembler or compiler.

SD=SMALLDATA/S As with SMALLCODE the section names are ignored, but now for all Data and Bss sections. Important: Data and Bss will not be mixed. This large section will contain first all Data and then all Bss sections.
Because the Bss part has no definite contents (only zeros), only the Data part will be stored. The size of Bss is stored together with the Data size in the load file's header.
Since OS2.0 the Bss part will be initialized to zero, when it's loaded (e.g. by LoadSegment()). But beware (!!), this is not the case with Kickstart 1.x!
If you don't want to see your programs crash on some Kick 1.3 dinosaurs, I recommend to clear the Bss part manually by using the special Linker-symbols _DATA_BAS_, _DATA_LEN_ and _BSS_LEN_ (refer to Linker Symbols for more information).
All references to symbols of this Small Data section will be calculated as if the Bss sections were directly behind the Data.
You should use this switch, when compiling/assembling your code with Small Data model enabled.

ND=NODEBUG/S The HUNK_SYMBOL and HUNK_DEBUG blocks, which contain informations for a debugger will not be included in the output file.

PV=PVCOMPAT/S This switch activates the PowerVisor compatibility mode, which is necassary when using Source Level Debugging informations in your program. Unfortunately the author of PowerVisor, Jorrit Tyberghein, currently does not plan a new release.

MAXSECTS/K/N Determines the maximum number of sections per unit. The default value ist 16, which should be enough for most cases.

BUFSIZE/K/N Changes the size of the buffer, required for buffered I/O. The buffer size defaults to 8192 bytes.

DEF=DEFINE/K Defines a Linker Symbol, which has a higher priority than all symbols of the same name which follow. Definition of multiple symbols must be seperated by commas. Don't forget to embed the whole term which follows DEFINE, in quotes (because of some problems with ReadArgs()) !
You may assign an absolute number or an absolute or relocatible symbol. This makes it possible to redirect calls to specific library functions to your own routines, etc..

HT=HASHTAB/K/N Number of entries in the hash table for XDEF symbols as a power of 2. It is valid from 2^8 to 2^16. The default value is 2^10.

NOSHORTRELOCS/S Don't generate RELOC32SHORT hunk blocks.

DONTSHORTENSECT/S Don't delete zero bytes at the end of each section.

BATCH/S
ADDSYM/S
NOALVS/S
NOICONS/S
LIB/S
These five switches have absolutely no effect. They were implemented to improve compatibility with BLink and SLink and to minimize difficulties with Makefiles.

ALV/S Allows PhxLnk to generate Automatic Link Vectors (ALV), when required. By default, generation of ALVs is disabled to guarantee compatibility with older PhxLnk versions (ALV/S exists since v4.30), and, because they are not always wanted and not without any danger.
PC-relative 16-bit references, which point to a symbol residing in a different section, and which should normally lead to an error message, will be redirected to an artificially generated jump table.
It is dangerous, because it's impossible to determine, whether the 16-bit reference is a jump instruction or a data access.

1.8 Linker Symbols

```
The linker itself creates some ext_def($01xxxxxx) and ext_abs($02xxxxxx)
symbols which will be needed by the startup code of a program using the
Small Data model.

_DATA_BAS_       (ext_def) Base address of the small data section.
_CODE_LEN_       (ext_abs) Length of the small code section in bytes.
_DATA_LEN_       (ext_abs) Length of the Data-part of the small data section.
_BSS_LEN_        (ext_abs) Length of the Bss-part of the small data section.
_SMALL_DATA_LEN_ (ext_abs) _DATA_LEN_+_BSS_LEN_ (zero in large data mode)

If the small code or small data mode was not activated, the lengths
always refer two the first section of every type.

For compatibility with Lattice/SAS or DICE you may also use these symbols:

Lattice/SAS:
_LinkerDB   (ext_def) This symbol can be used to initialize your small
            data base register. Normally it will point 32766 bytes into the
            small data section, but when the BLink compatibility switch
            was selected and the small data area is smaller than 32k,
            it will point to its beginning.
__BSSBAS    (ext_def) Base address of the Bss-part of the small data
            section.
__BSSLEN    (ext_abs) Length of the Bss-part in longwords.
___ctors and ___dtors are always zero.

DICE:
__DATA_BAS  (ext_def) Base address of the small data section.
__DATA_LEN  (ext_abs) Length of the Data-part of the small data section in
            longwords.
__BSS_LEN   (ext_abs) Length of the Bss-part of the small data section in
            longwords.
__RESIDENT  (ext_abs) Always zero.
```

1.9 Small Data

```
small data symbols can be accessed in a range of 65534 ($fffe) bytes. When
a symbol is outside of this range, the linker will display an error.
The small data model must be initialized by the startup code. When you're
using A4 as small data pointer, the initialization would look like this:

 xref    _DATA_BAS_             ; _DATA_BAS_ is a linker symbol

 lea     _DATA_BAS_+32766,a4    ; a4 always points to the mid. of small data
```

1.10 Known Bugs

```
o If the output file has more than 1000 sections there could be a
  stack overflow :-)
```

If any bugs or questions occur, please write to :

My Address

1.11 My Address

Snail Mail:

Frank Wille
Auf dem Dreische 45
32049 Herford
GERMANY

Electronic Mail:

frank@phoenix.owl.de

World Wide Web:

http://gaia.owl.de/~frank/phxlnk_e.html

```
   _
_ //
\X/  A M I G A   F O R E V E R  !
```