PhxLnk_d

COLLABORATORS

	TITLE : PhxLnk_d	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 1, 2022	

REVISION HISTORY

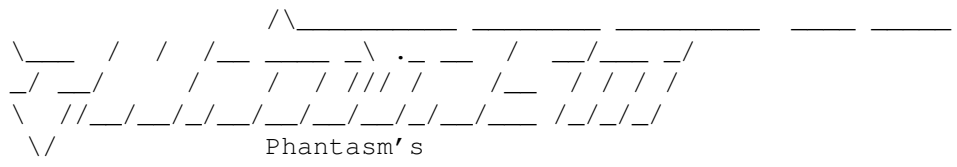
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PhxLnk_d

1.1 PhxLnk V4.23 (03.05.96)



P h x L n k V4.23

AMIGA-DOS Module Linker

Inhalt

1.2 Einleitung

PhxLnk wurde, unter Verwendung des Assemblers PhxAss, in reinem Assembler-Code geschrieben. Die erste Version wurde mit BLink erzeugt, für die nachfolgenden wurde jeweils die letzte PhxLnk-Version verwendet.

PhxLnk unterstützt alle Features eines Standard Amiga-DOS Linkers, wie z.B. BLink, bis auf Overlay Hunks und Automatic Link Vectors (ALVs).

Barthelmann.
Die kommerzielle Nutzung dieses Programms ist strikt untersagt!

1.3 Verbesserungen seit V3.00

V4.23 (03.05.96) LoadSeg() V40 hat offensichtlich noch ↩
Probleme mit
einer Section-Länge von Null, da in diesem Spezial-
fall der möglicherweise vorhandene BSS-Teil einer
Section nicht mehr gelöscht wird. Eine minimale
Section-Länge von einem Longword ist von nun an
garantiert.
Anstelle der dos.library Buffered I/O Routinen
werden eigene, schnellere, benutzt. Die Größe des
Buffers läßt sich über den neuen CLI-Parameter

BUFSIZE/K/N
variieren und ist auf 8192 vorein-
gestellt.
Das Erzeugen einer FFS-Datei ist mehr als fünfmal
so schnell wie mit v4.22.

V4.22 (23.03.96) Die maximal erlaubte Länge des Zieldateinamens
war deutlich zu gering dimensioniert und liegt
jetzt bei 233.
Der XREF-Subtyp EXT_RELREF32 ($88) für 32-Bit
PC-Relative Referenzen wird unterstützt.
Beim Erzeugen der Zieldatei wird nun Buffered I/O
(FWrite) verwendet.

V4.21 (08.03.96) PhxLnk hatte Probleme mit mehreren Punkten in einem
Dateinamen. Ein Object File mit dem Namen "a.b.c.o"
wurde dadurch zu "a" statt "a.b.c".
Die Ausgabe der undefinierten Symbole läßt sich
nun jederzeit mit CTRL-C abbrechen.

V4.20 (20.12.95) Seit v4.20 gibt es zwei Versionen von PhxLnk.
Eine für OS2.x und die andere für OS3.x. Diese
Aufteilung ist ein Resultat der intensiven Nut-
zung der unter OS3.x in der exec.library vor-
handenen MemPool Funktionen (dieser werden in
der OS2.x-Version aus der amiga.lib hinzugezo-
gen). Die Umstellung auf MemPools stammt von
Volker Barthelmann <volker@vb.franken.de>. Sie
ermöglicht eine vielfache Geschwindigkeitsstei-
gerung beim Linken mit Libraries.
Es kam teilweise zu Abstürzen, wenn leere Sec-
tionen, die allerdings einige benötigte XDEFs
enthielten, automatisch gelöscht wurde. Dies
ließ sich nur duch Setzen des
PRESERVE/S
Schalters vermeiden.

V4.17 (24.05.95) Der SLink aus dem SAS/C-Paket wird nun teilweise
unterstützt, indem PhxLnk die Spezial-Symbole

___ctors und __dtors als NULL definiert. Vollständige SLink-Unterstützung wird vielleicht in einer der nächsten Versionen folgen. (wenn dies jemand benötigen sollte :-)

V4.16 (18.05.95) Durch die Korrektur des __MERGED-Bugs in V4.10 funktionierte der 'normale' Small Data Modus nicht mehr.
Das Entfernen von Nullbytes am Ende einer Section funktionierte noch nicht so gut wie es sollte.

V4.15 (19.03.95) PhxLnk kann RELOC32SHORT-Blöcke erzeugen.

V4.10 (21.02.95) Das Linken mit __MERGED-Sections hatte noch einen Fehler.

V4.03 (09.02.95) Ein Read-Error, während PhxLnk eine '@'-Datei liest, führte zu einem Absturz.

V4.02 (25.01.95) Die Namen der zu linkenden Objekte können auch aus einer oder mehreren Dateien gelesen werden.

V4.00 (18.11.94) PhxLnk V4.00 benötigt mindestens OS2.04. Dadurch ist das Programm um einiges kürzer geworden und benutzt jetzt Standard Command Line Parsing via ReadArgs(). DEBUG Hunk Blöcke werden jetzt genauso behandelt wie bei BLink. Dadurch ist es ohne weiteres möglich Load Files für einen Source Level Debugger zu generieren. Die neueste Version des PowerVisor Debuggrs, V1.42, hat aber leider einige Probleme mit Programmen die von BLink oder PhxLnk erzeugt worden sind, was unter anderem Source Level Debugging mit mehreren verschiedenen Source Files praktisch unmöglich macht.
Um dies zu beheben, wurde ein PowerVisor Kompatibilitäts-Schalter,
PVCOMPAT
, eingeführt.
Durch das neue Argument DEFINE (siehe
Parameter
)
ist die Definition von absoluten Linker-Symbolen möglich (so ähnlich wie die Linker-Symbole die im Small-Data Modus benötigt werden).
Durch den Schalter
BLINKCOMPAT
verhält sich
PhxLnk beim Linken von Small-Data Modulen wie der BLink von SAS/C.
Data- oder Bss-Sections, die den Namen "__MERGED" tragen, werden automatisch zu einer Small-Data Section verbunden (ohne den SMALLDATA-Schalter).
Außerdem gibt es neben der englischen nun auch eine deutsche Anleitung.
Bei Code- oder Data-Sections werden Null-Bytes am Ende der Section nicht mehr mit gespeichert, sondern nur noch im HUNK_HEADER vermerkt. Da dies unter Kickstart 1.x nicht möglich ist, gibt es den

Kompatibilitätsschalter
KICK1
.

V3.10 (04.08.94) Katastrophalen Fehler beseitigt, der manchmal beim Linken von Libraries auftaucht.
Um ehrlich zu sein: Ich glaube nicht, daß eine von den bisherigen PhxLnk-Versionen fehlerlos genug war, um Libraries wirklich sicher zu Linken.
PhxLnk wurde vollkommen lokalisiert. Bis jetzt sind deutsche und polnische Kataloge verfügbar.
Die Anleitung liegt jetzt im Amiga-Guide Format vor.

V3.05 (31.07.94) Wieder mal einen Linker-Library Bug berichtigt: Manchmal passierte es, das Sectionen einer Library, obwohl der Linker eigentlich entschieden hatte sie nicht zum Load File hinzuzufügen, trotzdem im HUNK_HEADER mit einer zufälligen Längenangabe erschienen.
HUNK_RELOC und HUNK_SYMBOL Blöcke der Länge Null, werden nicht mehr beachtet.

V3.01 (22.01.94) Durch die massiven Änderungen im V3.00 hatte sich ein kleiner Fehler mit dem Namen der Zieldatei eingeschlichen.

V3.00 (18.01.94) Mehrere Probleme beim Linken von Libraries korrigiert, die z.B. zu einem FreeMemoryTwice Guru (oder Schlimmerem) führten.
Die wichtigsten Linker-Symbole von Lattice/SAS (_LinkerDB,__BSSBAS,__BSSLEN) und DICE (__RESIDENT, __DATA_BAS,__DATA_LEN,__BSS_LEN) werden nun ebenfalls unterstützt.
PhxLnk kann das spezielle Library-Format von SAS/C, in dem es in das Standard-Format übersetzt wird, lesen.

1.4 Benutzung von PhxLnk

Systemanforderungen

PhxLnk starten

Parameter

Linker Symbole

Small Data

1.5 Sytemanforderungen

Seit PhxLnk V4.00 ist das Betriebssystem OS2.04 (V37) absolutes Minimum. Dadurch ist PhxLnk viel kürzer, schneller und leichter für mich zu programmieren. Ich glaube auch nicht, daß diese Einschränkung (die in meiner Ansicht nach ein Fortschritt ist) heutzutage noch irgendjemanden trifft.

PhxLnk wurde mit den folgenden Konfigurationen getestet:
A4000(68040), 2 Chip, 16 Fast, OS3.1
A1000(68010), 0.5 Chip, 2 Fast, OS3.1
A500(68EC030), 1 Chip, 1 Fast, OS3.1

1.6 PhxLnk starten

PhxLnk wird normalerweise von der Shell aus gestartet. Dazu sollten sie PhxLnk entweder aus dem OS2.x-Ordner, falls Sie OS2.x fahren, oder dem OS3.x-Ordner, falls OS3.x, nach C: kopieren, oder wenigstens einen Pfad oder Link definieren. Die OS2.x-Version läuft auch auf OS3.x-Amigas, ist aber etwas länger.

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

Schablone: 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,DEF=DEFINE/K
```

Wird PhxLnk ohne jedes Argument oder mit einem einfachen '?' aufgerufen, so wird eine kurze Beschreibung der möglichen Parameter ausgegeben. Für eine genauere Beschreibung der Parameter verweise ich auf das Kapitel
Parameter
.

Es gibt drei Arten von Modulen die gelinkt werden können:
- Object Module mit Namenserweiterung ".o" oder ".obj", die normalerweise aus nur einer Unit bestehen. PhxLnk linkt aber auch Object Module mit mehreren Units.
- Library Module mit Namenserweiterung ".lib" bestehen normalerweise aus einer großen Anzahl Units, von denen PhxLnk nur die mit einbinden wird, wo mindestens ein ext_def-Symbol von einem anderen Modul benötigt wird.
- Lattice/SAS Extended Library Module (ebenfalls mit ".lib" Namenserweiterung). Sie werden von PhxLnk in das Standard Library Format übersetzt (keine besonders gute Lösung - aber es funktioniert).

Alle anderen Namenserweiterungen werden sofort zurückgewiesen.

Die Modulnamen können in jeder beliebiger Reihenfolge erscheinen, unter der Vorraussetzung, daß das erste ein Object Modul ist, welches die Start-Routine des Programms enthält.

WICHTIG!
Die von PhxLnk erzeugten Load Files sind standardmäßig NICHT Kickstart 1.x kompatibel! PhxLnk löscht Null-Bytes am Ende einer Section und versucht

die viel kürzeren RELOC32SHORT Blöcke zu verwenden, solange er nicht
duch setzen des
KICK1-Schalters
davon abgehalten wird.

1.7 Beschreibung aller Parameter

FROM/M Hierzu gehören unter anderem alle Parameter, die ↩
keinem der
anderen Schlüsselworte zugeordnet werden konnten. Sie stehen
für die Namen der Object und Library Module die gelinkt
werden sollen. Alle erlaubten Namenserweiterungen dieser
Module sind in
PhxLnk starten
aufgeführt.
Wenn ein Name mit '@' beginnt, so ist dies der Name einer
Datei, die weitere Objektnamen enthalten kann (sogar weitere
'@'). Die Trennung der Namen kann durch Leerzeichen, TABs,
LineFeeds oder sonstiges erfolgen. '"' werden unterstützt.

TO/K Bestimmt den Namen der zu produzierenden Zieldatei. Wenn keine angegeben wurde, hat die Zieldatei den Namen des ersten Moduls ohne Namenserweiterung. Beispiel: "PhxLnk prog1.o prog2.o c.lib m.lib" würde eine Load File mit dem Namen "prog1" generieren.

CHIP/S Durch diesen Schalter werden alle Sectionen beim Start ins Chip Ram geladen.

PRESERVE/S Der Normalfall ist, daß PhxLnk alle Sectionen der Länge Null entfernt um Speicherplatz zu sparen. Diesen Schalter sollten Sie dann wählen, wenn Sie die leeren Sectionen erhalten möchten.

B=BLINKCOMPAT/S PhxLnk verhält sich beim Linken von Small-Data Modulen kompatibel zu SAS/C's BLink. Das heißt, wenn die Small-Data Section kleiner als 32k ist, zeigt der Pointer (_LinkerDB) im Small-Data Basisregister auf den Sections-Anfang, statt 32766 Bytes in die Section hinein. Somit beginnen die Offsets bei 0 anstatt bei -32766.

SC=SMALLCODE/S Normalerweise werden beim Linken nur die Sectionen vereinigt, die vom selben Typ sind und den gleichen Namen haben. Durch diesen Schalter werden die Namen der Code Sectionen ignoriert, wodurch eine große Code Section entsteht.
SMALLCODE wird z.B. immer dann gewählt, wenn auch Ihr Assembler oder Compiler den Small Code Modus aktiviert hatte.

SD=SMALLDATA/S Hierdurch lassen sich sämtliche Data und Bss Sectionen zu einer großen Small Data Section vereinigen, dabei werden die Data und Bss Sectionen allerdings nicht wild gemischt. Die Small Data Section enthält zuerst alle Data Sectionen und dann alle Bss.
Da der Bss-Teil keinen bestimmten Inhalt hat (nur 0-Bytes), braucht auch nur der Data-Teil gespeichert zu werden. Die

Größe des Bss-Teils wird zusammen mit der des Data-Teils im Header des Load Files vermerkt.
Seit OS2.0 wird der nichtinitialisierte Bss-Teil beim Laden (z.B. mit LoadSegment()) automatisch mit Nullen gefüllt. Aber Vorsicht(!), dies ist nicht der Fall mit Kickstart 1.x! Wenn Sie Ihr Programm nicht auf irgendeinem dieser Kick 1.3 Dinosaurier abstürzen sehen wollen, empfehle ich den Bss-Teil manuell, d.h. beim Programmstart, durch Verwendung der dafür gedachten Linker-Symbole _DATA_BAS_, _DATA_LEN_ und _BSS_LEN_ zu löschen (siehe
Linker Symbole
).
Alle Zugriffe auf Small Data Symbole werden so berechnet, als wenn die Bss Sectionen direkt hinter den Data Sectionen wären. Diesen Schalter sollten Sie benutzen, wenn Sie Ihr Programm vorher im
Small Data
Modus übersetzt haben.

ND=NODEBUG/S — Alle HUNK_SYMBOL und HUNK_DEBUG Blöcke, die Informationen für einen Debugger beinhalten, werden nicht mit in die Zieldatei aufgenommen.

PV=PVCOMPAT/S — Dieser Schalter aktiviert den PowerVisor Kompatiblitäts-Modus, der erforderlich ist wenn Sie Source Level Debugging Informationen in Ihrem Programm haben. Leider plant der Autor von PowerVisor, Jorrit Tyberghein, momentan keine neue Version.

K1=KICK1/S — PhxLnk erzeugt Kickstart 1.x kompatible Load Files. Dieser Schalter verhindert das Löschen von Null-Bytes am Ende einer Section sowie die Nutzung von RELOC32SHORT-Blöcken.

MAXSECTS/K/N — Bestimmt die maximale Anzahl Sectionen pro Unit. Voreingestellt ist 16, was für die meisten Fälle auch reichen dürfte.

BUFSIZE/K/N — Ändert die Größe des Buffers, welcher für die Buffered I/O Funktionen benötigt wird. Voreingestellt ist eine Buffer-Größe von 8192 Bytes.

DEF=DEFINE/K — Definiert ein absolutes Linker-Symbol. Wenn Sie mehrere Symbole definieren wollen, müssen Sie sie durch ein Komma voneinander trennen. Der gesamte Definitionsteil hinter DEFINE sollte immer von Anführungszeichen eingeschlossen sein (da ReadArgs() sonst Probleme bekommt).

1.8 Linker Symbole

Der Linker selbst erzeugt einige ext_def($01xxxxxx) und ext_abs(↩
$02xxxxxx)
Symbole, die vom Startup Code eines Programms benötigt werden das im Small Data Modus übersetzt worden ist.

```
_DATA_BAS_  (ext_def) Basisadresse der Small Data Section.
_DATA_LEN_  (ext_abs) Länge des Data-Teils der Small Data Section.
_BSS_LEN_   (ext_abs) Länge des Bss-Teils der Small Data Section.


Um Kompatibilität mit Lattice/SAS oder DICE zu gewährleisten, werden auch
folgende Linker Symbole unterstützt:

Lattice/SAS:
_LinkerDB   (ext_def) Dieses Symbol kann direkt dazu benutzt werden das
            Small-Data Basisregister zu initialisieren. Normalerweise
            zeigt _LinkerDB 32766 Bytes in die Small-Data Section hinein,
            doch durch Setzen des
                BLink-Kompatibilitätsschalters
                 ist es
            möglich, daß es auf den Anfang der Section zeigt, falls diese
            kleiner als 32k ist.
__BSSBAS    (ext_def) Basisadresse des Bss-Teils der Small Data Section.
__BSSLEN    (ext_abs) Länge des Bss-Teils in Longwords.
___ctors und ___dtors sind immer NULL.

DICE:
__DATA_BAS  (ext_def) Basisadresse der Small Data Section.
__DATA_LEN  (ext_abs) Länge des Data-Teils in Longwords.
__BSS_LEN   (ext_abs) Länge des Bss-Teils in Longwords.
__RESIDENT  (ext_abs) Immer Null.
```

1.9 Small Data

```
                Small Data Symbole können in einem Bereich von 65534 ($fffe) Bytes ↩
                    adres-
siert werden. Wenn der Linker feststellt, daß sich ein Symbol außerhalb
dieses Bereichs befindet, wird eine Fehlermeldung ausgegeben.
Der Small Data Modus erfordert eine Initialisierung beim Programmstart. Wenn
Sie sich z.B. dafür entschieden haben, A4 als Small Data Basisregister zu
verwenden, würde die Initialisierung folgendermaßen aussehen:

 xref    _DATA_BAS_             ;
                _DATA_BAS_
                 ist ein Linker Symbol

 lea     _DATA_BAS_+32766,a4    ; a4 zeigt immer in die Mitte von Small Data
```

1.10 Fehler

```
                o Wenn die Zieldatei mehr als 1000 Sectionen hat, könnte es zu  ↩
                   einem
  Stack Overflow kommen :-)


Wenn noch irgendwelche Fragen oder Fehler auftauchen, schreiben Sie an:
```

Meine Adresse

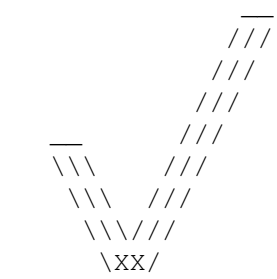
1.11 Meine Adresse

Standard Mail:

Frank Wille
Auf dem Dreische 45
32049 Herford
DEUTSCHLAND

Electronic Mail:

frank@phoenix.owl.de



A M I G A F O R E V E R !