Cereon

CDS 1.0
Linker reference

Authors:
Andrei Kapustin

Revision history

Date	Comment
30 Sep 2008	Initial draft.

1 Contents

2 Introduction

This document is a definitive guide into the Cereon Linker (henceforth referred to simply as an “linker”).

The Cereon Linker is a tool that creates executable files, which can then be executed on a Cereon platform, and dynamic-link libraries, which can then be used by dynamic linker to resolve external references at load-time and/or run-time.

Cereon Linker uses NGOFF (Next Generation Object File Format) as its primary object, executable and dynamic-link library format and NGOLF (Next Generation Object Library Format) as its primary object library and import library format.

3 The linking process

The Cereon Linker takes as an input a set of object files and / or object libraries, producing the executable file (which can be either application or dynamic-link library, depending on linker's command-line options).

3.1 Long branch veneers

This section describes the technique used by linker known as "long branch veneers". It is used to allow jump and branch instructions to transfer control to any location within the 64-bit address space, even though the actual Cereon instructions that implement jumps and branches have limited range (specifically, a branch instruction can directly transfer control to any other instruction within the $[-2^{15}..2^{15}-1]$ instructions range, while jumps are limited to a somewhat larger range of $[-2^{23}..2^{23}-1]$ instructions).

Note that long branch veneers are not specific to the assembler language. Compilers from high-level languages that target Cereon platforms will routinely assume the linker's ability to generate long branch veneers when generating code (specifically procedure call/return instruction sequences).

In essence, when a jump or branch instruction with an implicit base (such as "`j x`" or "`beq.l $t0, $t1, x`") is translated and the jump target `x` is not within a range that can be guaranteed to be reachable for the jump or branch in question (for example, when `x` is an external symbol, or resides in a different section), the jump or branch instruction is generated without a pre-calculated jump offset, but with an appropriate relocation; it is also marked as being a jump or branch.

When linker later creates an image, it has two choices:

- If, in the generated image, the jump target `x` is reachable directly, the jump or branch instruction is relocated as necessary to do so (for example, the external jump target `x` may have ended up in the same section as the jump or branch instruction that uses it due to section joining). In this case, there is no overhead involved; the jump or branch instruction will transfer control directly to the jump target.
- If, in the generated image, the jump target `x` cannot be safely reached by the jump or branch instruction, a long branch veneer is generated as described below.

A long branch veneer is a snippet of code that is:

- Automatically generated by the linker if necessary,
- Is guaranteed to be placed within a reachable range of the jump or branch instruction that uses it, and
- Can transfer control anywhere within the 64-bit address space.

For example, consider an assembler language program that consists of one code section and has, within that code section, a jump instruction that transfers control to an external symbol:

```
        import  proc
.text:  section code
        jal     proc
        halt
.text:  end section
```

The actual implementation of the `proc` routine is, in this case, imported from a dynamic-link library, which means that its address at runtime is more or less arbitrary. After linking, the image will contain the following code:

```
        import  proc
.text:  section code
        jal     proc::veneer
        halt
        align   8
proc::address:
        long    proc
proc::veneer:
        lir     $ip, proc::address
.text:  end section
```

What happened here is the jump-and-link instruction being re-targeted to the veneer, which loads a full 64-bit address of the jump target directly into `$ip`. When the `proc` procedure performs return (by executing the "`jr $ra`" instruction), the control is transferred back to the instruction following the `jal`, just as if `jal` was able to jump to the `proc` entry point directly.

Depending on the code alignment, each long branch veneer is either 12 or 16 bytes long. Naturally, the linker will try to minimize the total number of veneers it generates so as to keep the code size to a minimum. Note, however, that veneers can only be inserted at section boundaries; therefore if you have a 512KB section with a conditional branch to an external label right in the middle, there will be no way for linker to generate a veneer, as both section boundaries are far beyond the reach of a conditional branch instruction.

Note also that the apparent loading of a jump target address from a `code` (i.e. executable but not readable as data) section is no threat, as the Cereon `lir` instruction has been specifically designed for this purpose – among other things, it required "execute", not "read", memory access permission for the loading to succeed. If the instruction "`li.l $ip, -12[$ip]`" was used instead, the example above would not work, as loading the data from an execute-only section by "`li.l`" would have caused a `DACCESS` (data access denied) exception instead.

The linker will also perform two more optimizations to further reduce the veneer size – veneer pairing and veneer chaining.

Veneer pairing comes in handy when several consecutive veneers must be generated, as in:

```
        import  proc1
        import  proc2
.text:  section code
        jal     proc1
        jal     proc2
        halt
.text:  end section
```

Normally, the linked image would contain the following code:

```
        import  proc1
        import  proc2
.text:  section code
        jal     proc1::veneer
        jal     proc2::veneer
        halt
        align   8
proc1::address:
        long    proc
proc1::veneer:
        lir     $ip, proc1::address
        align   8
proc2::address:
        long    proc
proc2::veneer:
        lir     $ip, proc2::address
.text:  end section
```

Note that the second alignment directive "`align 8`" would introduce 4 bytes of padding between veneers, as each veneer is 12 bytes long yet must be aligned on a 8-byte boundary. With veneer pairing, the 2nd veneer of each pair is reversed in order, giving:

```
        import  proc1
        import  proc2
.text:  section code
        jal     proc1::veneer
        jal     proc2::veneer
        halt
        align   8
proc1::address:
        long    proc
proc1::veneer:
        lir     $ip, proc1::address
proc2::veneer:
        lir     $ip, proc2::address
proc2::address:
        long    proc
```

```
.text:  end section
```

Now the 4 padding bytes between veneers are gone, as `proc2::address` is guaranteed to be 8-byte-aligned.

The veneer chaining comes in handy when several veneers must be generated for jump targets that are at a fixed offset of each other. Consider:

```
        extern  proc1
        extern  proc2
.text:  section code
        jal     proc1
        jal     proc2
        halt
.text:  end section
```

Now, suppose that the linker determines that both `proc1` and `proc2` are out of jump range, but also that `proc2`'s address is exactly 200 bytes larger that `proc1`'s address. While the direct approach is possible, as in:

```
        import  proc1
        import  proc2
.text:  section code
        jal     proc1::veneer
        jal     proc2::veneer
        halt
        align   8
proc1::address:
        long    proc
proc1::veneer:
        lir     $ip, proc1::address
proc2::veneer:
        lir     $ip, proc2::address
proc2::address:
        long    proc
.text:  end section
```

the shorter alternative would also work:

```
        extern  proc1
        extern  proc2
.text:  section code
        jal     proc1::veneer
        jal     proc2::veneer
        halt
        align   8
proc1::address:
        long    proc
proc1::veneer:
        lir     $ip, proc1::address
```

```
proc2::veneer:
        lir     $ra, proc1::address
        addi.l  $ip, $ra, 200
.text:  end section
```

The veneer code is now 20 bytes long instead of 24. The use of `$ra` as a temporary register is safe, because `$ra` is not expected to keep its value across procedure calls or be initialized when `proc2` is called. Naturally, for this kind of optimization to be applicable, the linker must be informed that one of the standard Cereon procedure calling standards is in use.

3.2 Roaming section handling

A "roaming" section is a section defined by an object module given to a linker as input that is not assigned to a particular segment (as opposed to "bound" sections, which are pre-assigned to segments). The linker will automatically generate segments to assign roaming sections to according to the following rules:

- A roaming section that is assigned an address will be placed into its own auto-generated segment with the same address and the same access rights as the section itself.
- All roaming sections not assigned an address will be collected into auto-generated segments according to their access rights (e.g. all "read-only" roaming sections will be collected into a read-only data segment, all "read-write" roaming sections will be collected into a read-write data segment, etc.) As there are three access permissions a section can have ("read", "write" and "execute"), at most 8 segments can be auto-generated for these sections.
- Any two adjoining auto-generated segments with the same access permissions will be merged. For example, if the input object modules specify two read-only roaming sections at addresses `0x0000000000000000` and `0x0000000000000010`, each 16 bytes long, these sections will be assigned to two auto-generated read-only segments at the same addresses, which would then be joined into a single read-only segment 32 bytes long at address `0x0000000000000000`.
- Any two auto-generated segments not assigned an address but having compatible access rights will be merged. Note that the notion of "compatible access rights" depends on the selected target platform. For example, when linking for Cereon-1P1B target, the linker "knows" that the entire address range has the "read" permission, so if two segments are auto-generated to host roaming sections – one with "read/write" access permissions and one with "write" access permissions, these two will be joined.

All in all, the linker tries to reduce the number of auto-generated segments hosting roaming sections to a minimum while preserving section properties.

3.3 Linking with DLLs

In addition to basic linking functionality, the linker supports dynamic loading by allowing the following inputs to the linking process:

- Import libraries – when a dynamic-link library is created, the corresponding "import library" is usually created as well. The "import library" is a static library that the linker to resolve references to global symbols defined by the dynamic-link library at load-time.
- For some input formats (e.g. NGOFF), the dynamic-link library itself can be specified as a linker input file. In this case the corresponding import library is generated behind-the-scenes and used as if it was explicitly given to the linker. To the programmer, things look like the linker resolves external references directly against the dynamic-link library.

Note that in both cases the dynamic-link library in question will be linked against at application load time. Depending on the target OS, dynamic linking at run time may or may not be supported as well; the linker is not responsible for that functionality.

You can use dynamic loading when linking both applications and dynamic-link libraries.

4 Invoking the linker

The general form of linker invocation is:

```
cerlink [ -<option> ...] [--] <filename>...
```

where:

- `cerlink` is the name of the linker executable. Depending on the host OS the actual name of the linker executable file may be different (for example, on Windows it is `cerlink.exe`).
- Each `-<option>` specifies a command line option that adjusts some facet of the linker behaviour.
- Each `<filename>` must be a name of an object file or an object library (the recommended filename extension for object files is `.rel`, for "RELocatable"; the recommended filename extension for object library files is `.stl`, an abbreviation from "STatic Library").
- An option terminator (two consecutive dashes `--`), if present in the command line as an independent option, marks the end of the options list; all subsequent parameters are treated as object file and/or object library names even if they start with a dash.

Note that, although object files and object library files are supposed to have different filename extensions, the linker's decision on whether an input file is an object file or an object library is made based on the actual contents of the file, not its extension.

4.1 Environment variables

During operation, linker uses a number of OS environment variables, all of which are described below.

4.1.1 CDS_1_0_LINKER_OPTIONS

If this environment variable is defined, its value must be a list of linker options in the same format as used in the linker command line during linker invocation.

When a linker processes the command line, all options specified by the `CDS_1_0_LINKER_OPTIONS` environment variable are processed as if they were explicitly specified in the linker's command line; the only exception to that rule is in that if some option defined by the `CDS_1_0_LINKER_OPTIONS` environment variable conflicts with another option explicitly specified in the linker command line, the former is ignored. This behaviour allows using the `CDS_1_0_LINKER_OPTIONS` environment variable to set the most commonly needed linker options, which can be overridden in each particular case if necessary.

For example, if the value of the `CDS_1_0_LINKER_OPTIONS` environment variable is "`-Endian:Big`", linker will always produce executables for a big-endian target unless invoked with an option "`-Endian:Little`".

4.1.2 CDS_1_0_LIB

This environment variable, if defined, must have as its value a list of directories where object files and libraries are looked for. Note that these directories have lower priority than those designated for lookup with corresponding command line options.

4.2 Options

A linker option has one of the following forms:

```
-<option keyword>
-<option keyword>:<option parameters>
```

Where `<option keyword>` is a keyword identifying the option and `<option parameters>`, if present, specifies additional information for the option.

Both option keywords and option parameters are generally case-insensitive except where character case matters (for example, when specifying file or directory names as option parameters on a Unix host, where file system is case-sensitive).

For some options both forms (i.e. with and without option parameters) are permitted; for example the "`-Map:<map file name>`" option, which tells the linker to produce a link map into the specified file, can also be written as simply "`-Map`", in which case the map file name is derived from the output file name by replacing its filename extension with `.map`.

4.3 Option files

Quite frequently a need arises to specify the same set of options for a large number of linker invocations. If these options are specific to a given linker project, the facilities provided by the `CDS_1_0_LINKER_OPTIONS` environment variable and related variables are insufficient (as they affect all linker invocations); instead, option files can be set up and used.

An option file is a text file that contains one or more linker options. To improve readability, line breaks and extra spaces are allowed between options in an arbitrary manner.

When invoking the linker, a dedicated `–Via:<option file name>` option is used to tell the linker to read an option file, treating all linker options specified therein as if they occurred in the linker command line. For example, if an option file is named `options` and contains the following lines:

```
-Endian:Big
-WarningsAsErrors
```

then invoking the linker with the following command line:

```
cerlink –Map –Via:options test.rel
```

has the same effect as:

```
cerlink –Map -Endian:Big -WarningsAsErrors test.rel
```

Any number of option files can be specified; the maximum length of an option file is not limited except by available memory.

A special case is an option file including other option files (the `-Via:<option file name>` command line option can, like any other option, occur within an option file). Such inclusion is permitted as long as option file dependencies do not cause an inclusion cycle.

4.4 Option conflicts

Unlike most existing linkers, the Cereon Linker is very strict about its command line. In particular, it is not permitted to specify two conflicting options in the same linker invocation, so the following will cause a command-line error:

```
cerlink –Map –Endian:Big –Endian:Little test.rel
```

Most existing linkers will allow similar invocations (i.e. with conflicting options), automatically resolving these conflicts in an arbitrary manner (e.g. some linkers may choose the 1st definition of a conflicting option, while other linkers will use the last definition), sometimes with a warning. It is, however, a firm belief of the Cereon Linker authors that conflicting command line options shall not be permitted at all, as any strategy used for automatic conflict resolution makes the invocation result dependent upon the order in which linker options are specified.

The only situation where option conflicts are resolved automatically is when options explicitly specified in the linker command line (or in option files included by a `–via` option that occurs in the linker command line) are in conflict with options specified by the `CDS_1_0_LINKER_OPTIONS` environment variable (or in option files included by a `–Via` option that occurs in one of these environment variables). In this case, options specified in the linker command line are always chosen; this behaviour permits explicit overriding of any implicit linker options.

4.5 Character encoding

Normally, linker produces all textual output (i.e. map files) using the “native” character encoding of the host OS. The definition of a “native” encoding is specific to a given OS (for example, on Linux hosts the “native” encoding is selected when a host is configured).

It is, however, possible to specify that a different encoding shall be used on a per-text-type basis (i.e. “an encoding for map files”, “an encoding for error and warning messages”, etc.)

4.6 Message files

All messages issued by the linker (i.e. banners, statistics, warnings, errors, etc.) are stored in a textual message file. In the minimum configuration the linker can operate entirely without message files, in which case built-in English Neutral messages are

issued. Additional message files can also be provided in a particular linker installation, containing the linker messages localized for a specific culture.

When invoking a linker, a dedicated command line option `–Locale:<locale ID>` can be used to specify the culture for which linker messages shall be tailored. For example, invoking linker with an option `–Locale:de` will cause all linker messages, including diagnostics, to be issued in German (provided, of course, that the corresponding message file is available).

Message files always use UTF-8 encoding.

5 Linker options

This section contains a complete description of each linker option.

The following notation is used when describing linker options:

`[ x ]` – The element `x` is optional.
`[ x …]` – The element `x` can be repeated 0 or more times.
`{ x | y }` – Either `x` or `y` can be chosen.

5.1 -Banner

This option has the following format:

```
-Banner[:{On|Off}]
```

When used, this option turns on (`-Banner:On`) or off (`-Banner:Off`) the printing of the version and copyright banner when the linker is invoked; a default form `-Banner` is equivalent to `-Banner:On`, which is also the default linker behaviour.

5.2 -Debug

This option has the following format:

```
-Debug[:{On|Off}]
```

When used, this option specifies whether the generated executable or dynamic-link library contain debug information (`-Debug:On`) or not (`-Debug:Off`); a default form `-Debug` is equivalent to `-Debug:On`. The default linker behaviour (when this option is not specified) is not to include debug information into the output.

5.3 -DisableWarning

This option has the following format:

```
-DisableWarning:<warning ID>[,<warning ID> …]
```

When used, this option causes warnings with the specified ID not to be issued. Any number of comma-separated warning IDs can be specified in the same `–DisableWarning` option; similarly, it is permitted to have more than one `–DisableWarning` option in the same command line.

5.4 -Dll

This option has the following format:

```
-Dll
```

When used, this option causes the linker to produce a dynamic-link library instead of an executable. The option can be specified at most once.

5.5 -EnableWarning

This option has the following format:

```
-EnableWarning:<warning ID>[,<warning ID> …]
```

When used, this option causes warnings with the specified ID to be issued. Any number of comma-separated warning IDs can be specified in the same `-EnableWarning` option; similarly, it is permitted to have more than one `-EnableWarning` option in the same command line.

5.6 -Endian

This option has the following format:

```
-Endian:{Big|Little}
```

When used, this option causes the linker to create a big-endian (`-Endian:Big`) or little-endian (`-Endian:Little`) executable (or dynamic-link library), correspondingly. If the endianness for an output file is not specified, that of the first input object module is used.

5.7 -Entry

This option has the following format:

```
-Entry:<symbol name>
```

When used, this option causes the linker to select the global symbol `<symbol name>` as an entry point. The following rules are used:

- If the generated output file has no entry point, the specified symbol becomes its entry point.
- If the generated output file has a single entry point and the name of that entry point matches the `<symbol name>`, no further action is taken.
- If the generated output file has a single entry point and the name of that entry point does not match the `<symbol name>`, the specified symbol becomes a new entry point and a warning is issued.
- If the generated output file has multiple entry points and the name of one of these entry points matches the `<symbol name>`, that entry point is selected and all others are ignored.
- If the generated output file has multiple entry points and none of these entry points match the `<symbol name>`, the specified symbol becomes a new entry point and all existing entry points are ignored with a warning.

The above rules allow:

- Specifying an entry point for an executable or dynamic-link library that would otherwise have none.
- Overriding a default entry point for an executable or dynamic-link library.

- Explicitly selecting one of alternative entry points for an executable or dynamic-link library at link time.

5.8 -Format

This option has the following format:

```
-Format:<output format>
```

When used, this option instructs the linker to emit the output (i.e. executable or dynamic-link library) in the specified format. Use the "`cerlink -Help:Formats`" command to list supported formats.

5.9 -FullMessagePaths

This option has the following format:

```
-FullMessagePaths[:{On|Off}]
```

When used, this option specifies whether source file names where error and warning messages are anchored shall be printed in fully qualified (`-FullMessagePaths:On`) or relative (`-FullMessagePaths:Off`) form; a default form `-FullMessagePaths` is equivalent to `-FullMessagePaths:On`. The default linker behaviour (when this option is not specified) is to print anchor file names in a short (relative) form.

5.10 -Heap

This option has the following format:

```
-Heap:<reserve>[,<commit>]
```

When used, this option causes the generated executable or dynamic-link library to be marked as requiring the `<reserve>` bytes of heap, of which the `<commit>` bytes will be initially committed. If `<commit>` is not specified, no heap is initially committed.

Both `<reserve>` and `<commit>` have the following form:

```
<amount>[<unit>]
```

where:

- `<amount>` is the number of units to reserve (or commit). It can be specified as decimal, binary, octal or hexadecimal integer value using the same format as is used by the Cereon Assembler for integer constants.
- `<unit>` is an optional suffix specifying the size of an allocation unit as `B` (byte), `K`(`1K = 1024B`), `M` (`1M = 1024K`), `G` (`1G = 1024M`) or `T` (`1T = 1024G`). If the unit is not specified, `B` is assumed.

If both <reserve> and <commit> are specified, <commit> cannot be larger than <reserve>.

This option can appear at most once in the linker command line. If it is absent, the linker behaves as if "-Heap:1G,64K" was in effect.

5.11 -Help

This option has the following format:

```
-Help[:<subject>]
```

When used, it causes the help on the specific subject to be printed. Possible <subject> choices are:

- -Help (no subject) – prints the linker command line & options reference.
- -Help:Targets – prints the list of supported target platforms.
- -Help:Formats – prints the list of supported output formats.
- -Help:Encodings – prints the list of supported character sets.
- -Help:Locales – prints the list of supported message locales.

5.12 -ImpLib

This option has the following format:

```
-ImpLib[:<destination>]
```

When used, this option causes an import library for the generated executable or dynamic-link library to be written (typically, this option will be used when linking dynamic-link libraries). An import library is s static library which, when used for linking another executable, causes that executable to link to an appropriate dynamic-link library at load time.

Depending on the exact form of the option's parameter, the link map file may be written to several different locations:

- If the <destination> parameter is not specified, the name of the import library file is derived from the name of the executable or dynamic-link library file by replacing the filename extension with .iml. The import library is written into the same directory where the generated executable or dynamic-link library.
- If the <destination> parameter is specified and ends with a path component separator, it is assumed to refer to a directory where the import library file shall be written; the name of the import library file is derived from the name of the executable or dynamic-link library file by replacing the filename extension with .iml. The destination directory is created automatically if it does not already exist. Non-absolute directory names are considered to be relative to the current working directory, not to the generated executable or dynamic-link library file's location.

- If the `<destination>` parameter is specified and does not end with a path component separator, it is assumed to refer to a file where the import library shall be written. Non-absolute file names are considered to be relative to the current working directory, not to the generated executable or dynamic-link library's location.

5.13 -LibPath

This option has the following format:

```
-LibPath:<directory list>
```

When used, this option specifies the list of directories where linker looks up for object and library files. The `<directory list>` is a list of full paths of one or more directories separated by a host OS – specific path list separator (`:` on Unix, `;` on Windows, etc.)

Any number of `-LibPath` options can be specified on the linker's command line.

Library directories specified with this option have higher priority than those specified with the `CDS_1_0_LIB` environment variable (if one is defined).

5.14 -Locale

This option has the following format:

```
-Locale:<locale ID>
```

When used, this option causes all linker messages and diagnostics to be written using the specified locale instead of the default locale. The `<locale ID>` can have one of the following forms:

- `-Locale:<language>`, where `<language>` is a 2-letter ISO-639 language code – causes linker messages to be written using the neutral locale for the specified language.
- `-Locale:<language>_<country>`, where `<language>` is a 2-letter ISO-639 language code and `<country>` is a 2-letter ISO-3166 country code – causes linker messages to be written using the culture locale for the specified language and country.
- `-Locale:<language>_<country>_<variant>`, where `<language>` is a 2-letter ISO-639 language code, `<country>` is a 2-letter ISO-3166 country code and `<variant>` is an arbitrary locale variant string – causes linker messages to be written using the full locale for the specified language, country and variant.
- `-Locale:Native` – causes linker messages to be written using the "native" locale of the host OS.
- `-Locale:Invariant` – causes linker messages to be written using an invariant (culture-independent) locale.

If this option is not specified, the `–Locale:Invariant` is assumed. An attempt to instruct the linker to produce output and messages using a locale for which the librarian message file is not available causes the locale's parent to be used; if there is no message file for that parent locale, then its parent is used in turn, etc; if all else fails, an invariant locale is used for linker output and messages.

5.15 -Locals

This option has the following format:

```
-Locals[:{On|Off}]
```

When used, this option specifies whether the generated executable or dynamic-link library will contain local (i.e. non-global) symbols (`-Locals:On`) or not (`-Locals:Off`); a default form `-Locals` is equivalent to `-Locals:On`. The default linker behaviour (when this option is not specified) is not to include local symbols into the output.

5.16 -Map

This option has the following format:

```
-Map[:<destination>]
```

When used, this option causes a textual file containing the link map of a generated executable or dynamic-link library to be written.

Link map files are emitted in the "native" character set for the host OS unless another character set is selected with `–MapEncoding` option.

Depending on the exact form of the option's parameter, the link map file may be written to several different locations:

- If the `<destination>` parameter is not specified, the name of the link map file is derived from the name of the executable or dynamic-link library file by replacing the filename extension with `.map`. The link map file is written into the same directory where the generated executable or dynamic-link library.
- If the `<destination>` parameter is specified and ends with a path component separator, it is assumed to refer to a directory where the link map file shall be written; the name of the link map file is derived from the name of the executable or dynamic-link library file by replacing the filename extension with `.map`. The destination directory is created automatically if it does not already exist. Non-absolute directory names are considered to be relative to the current working directory, not to the generated executable or dynamic-link library file's location.
- If the `<destination>` parameter is specified and does not end with a path component separator, it is assumed to refer to a file where the link map shall be written. Non-absolute file names are considered to be relative to the current working directory, not to the generated executable or dynamic-link library's location.

- If the <destination> parameter is a single dash '-', the link map is written to the standard output; in this case the character set used for the dependencies output is that selected for the standard output with the –StdoutEncoding option.

5.17 -MapEncoding

This option has the following format:

```
-MapEncoding:<encoding>
```

When used, this option causes link map files to be written using the specified encoding (character set). The <encoding> parameter must be one of the encoding names supported by the linker, use the "cerlink –Help:Encodings" command to list supported encodings.

If this option is not specified, link map files are written using the "native" character set of the host OS. The same effect can be achieved by using this option with a special native parameter, as in

```
-MapEncoding:Native
```

5.18 -Model

This option has the following format:

```
-Model:{ilp64|lp64|ip32}
```

When used, this option sets the memory model to be used for the generated executable or dynamic-link library file. If the option is not specified, the memory model of the first input object module is used.

5.19 -Output

This option has the following format:

```
-Output:<destination>
```

When used, this option causes the resulting executable or dynamic-link library file to be written to the specified destination.

Depending on the exact form of the option's parameter, the output file may be written to several different locations:

- If the <destination> parameter is not specified, the name of the output file is the same as the name of the first input file, with the filename extension (typically .rel) replaced by whatever extension is used by selected output type and format (e.g. .run for NGOFF executables, .dyl for NGOFF dynamic-link libraries, etc.)
- If the <destination> parameter is specified and ends with a path component separator, it is assumed to refer to a directory where the output file

shall be written; the name of the output file is the same as the name of the first input file, with the filename extension (typically `.rel`) replaced by whatever extension is used by selected output type and format (e.g. `.run` for NGOFF executables, `.dyl` for NGOFF dynamic-link libraries, etc.) Non-absolute directory names are considered to be relative to the current working directory, not to the first input file's location.

- If the `<destination>` parameter is specified and does not end with a path component separator, it is assumed to refer to an output file itself. Non-absolute file names are considered to be relative to the current working directory.

5.20 -Progress

This option has the following format:

```
-Progress[:{On|Off}]
```

When used, this option turns on (`-Progress:On`) or off (`-Progress:Off`) the printing of linker actions as they are performed; a default form `-Progress` is equivalent to `-Progress:On`, while the default linker behaviour is not to emit progress messages.

5.21 -Quiet

This option has the following format:

```
-Quiet[:{On|Off}]
```

The `–Quiet:On` option is a shortcut for the `–Banner:Off –Progress:Off` and `–Statistics:Off` option combination. Similarly, the `–Quiet:Off` option is a shortcut for the `–Banner:On –Progress:On` and `–Statistics:On` option combination.

5.22 -Stack

This option has the following format:

```
-Stack:<reserve>[,<commit>]
```

When used, this option causes the generated executable or dynamic-link library to be marked as requiring the `<reserve>` bytes of stack, of which the `<commit>` bytes will be initially committed. If `<commit>` is not specified, no stack is initially committed.

Both `<reserve>` and `<commit>` have the following form:

```
<amount>[<unit>]
```

where:

- <amount> is the number of units to reserve (or commit). It can be specified as decimal, binary, octal or hexadecimal integer value using the same format as is used by the Cereon Assembler for integer constants.
- <unit> is an optional suffix specifying the size of an allocation unit as B (byte), K(1K = 1024B), M (1M = 1024K), G (1G = 1024M) or T (1T = 1024G). If the unit is not specified, B is assumed.

If both <reserve> and <commit> are specified, <commit> cannot be larger than <reserve>.

This option can appear at most once in the linker command line. If it is absent, the linker behaves as if "-Stack:1M,64K" was in effect.

5.23 -Statistics

This option has the following format:

```
-Statistics[:{On|Off}]
```

When used, this option turns on (-Statistics:On) or off (-Statistics:Off) the printing of processing statistics at the end of linker invocation; a default form -Statistics is equivalent to -Statistics:On, which is also the default linker behaviour.

5.24 -StderrEncoding

This option has the following format:

```
-StderrEncoding:<encoding>
```

When used, this option causes linker to issue all error and warning messages using the specified encoding (character set). The <encoding> parameter must be one of the encoding names supported by the librarian, use the "cerlink –Help:Encodings" command to list supported encodings.

If this option is not specified, all error and warning messages are written using the "native" character set of the host OS. The same effect can be achieved by using this option with a special native parameter, as in

```
-StderrEncoding:Native
```

5.25 -StdoutEncoding

This option has the following format:

```
-StdoutEncoding:<encoding>
```

When used, this option causes linker to issue all standard output using the specified encoding (character set). The <encoding> parameter must be one of the encoding names supported by the librarian, use the "cerlink –Help:Encodings" command to list supported encodings.

If this option is not specified, all standard output written by the librarian uses the "native" character set of the host OS. The same effect can be achieved by using this option with a special `native` parameter, as in

```
-StdoutEncoding:Native
```

5.26 -Target

This option has the following format:

```
-Target:<target platform>
```

When used, this option instructs the linker to produce an executable or dynamic-link library for the specified target. Use the "`cerlink –Help:Targets`" command to list supported targets.

If the target is not explicitly specified from the command line, the linker assumes the most permissive target (i.e. all features enabled).

5.27 -Veneers

This option has the following format:

```
-Veneers[:{On|Off}]
```

When used, this option allows (`-Veneers:On`) or disallows (`-Veneers:Off`) the generation of long jump veneers; a default form `-Veneers` is equivalent to `-Veneers:On`, which is also the default linker behaviour.

5.28 -Via

This option has the following format:

```
-Via:<option file name>
```

When used, this option instructs the linker to read the specified option file and process all command line options contained therein as if they were specified directly in the linker's command line in place of a `–Via` option.

Option files are assumed to be text files that use a "native" character set of the host OS; however, it is possible to instruct the linker to use a different encoding for option files with the `–ViaEncoding` option.

Option files can contain within them references to other option files (i.e. `–Via` options are permitted within option files) as long as option file references do not form a cycle.

5.29 -ViaEncoding

This option has the following format:

```
-ViaEncoding:<encoding>
```

When used, this option causes linker to assume that all option files use the specified encoding (character set). The `<encoding>` parameter must be one of the encoding names supported by the linker, use the "`cerlink –Help:Encodings`" command to list supported encodings.

If this option is not specified, all option files are assumed to be using the "native" character set of the host OS. The same effect can be achieved by using this option with a special `native` parameter, as in

```
-ViaEncoding:Native
```

5.30 -WarningAsError

This option has the following format:

```
-WarningAsError:<warning ID>[,<warning ID> …]
```

When used, this option causes warnings with the specified ID to be treated as errors. Any number of comma-separated warning IDs can be specified in the same `–WarningAsError` option; similarly, it is permitted to have more than one `–WarningAsError` option in the same command line. In particular, a warning issued as an error causes the linker to exit with a nonzero exit code, signalling a linker error.

Note that treatment of certain (or all) warnings as errors does not affect warning filtering – a warning message simply becomes an error message *if* it is decided that a warning message must be issued.

5.31 -WarningAsWarning

This option has the following format:

```
-WarningAsWarning:<warning ID>[,<warning ID> …]
```

When used, this option causes warnings with the specified ID to *not* be treated as errors. Any number of comma-separated warning IDs can be specified in the same `–WarningAsWarning` option; similarly, it is permitted to have more than one `–WarningAsWarning` option in the same command line.

Note that treatment of certain (or all) warnings as errors does not affect warning filtering – a warning message simply becomes an error message *if* it is decided that a warning message must be issued.

5.32 -WarningLevel

This option has the following format:

```
-WarningLevel:{0|1|2|3|4}
```

When used, this option selects the specified warning level. Generally, the higher the warning level the more warnings are issued; warning level 0 causes all warnings to be suppressed, while warning level 4 causes all warnings to be reported.

If the option is not specified, the default warning level is 2.

5.33 -WarningsAsErrors

This option has the following format:

```
-WarningsAsErrors[:{On|Off}]
```

When used, this option turns on (`-WarningsAsErrors:On`) or off (`-WarningsAsErrors:Off`) the treatment of all warnings as errors; a default form `-WarningsAsErrors` is equivalent to `-WarningsAsErrors:On`, while `-WarningsAsErrors:Off` is the linker default. In particular, a warning issued as an error causes the librarian to exit with a nonzero exit code, signalling a linking error.

Note that treatment of certain (or all) warnings as errors does not affect warning filtering – a warning message simply becomes an error message *if* it is decided that a warning message must be issued.

6 Appendix A: GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in

part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- **A.** Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- **B.** List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- **C.** State on the Title page the name of the publisher of the Modified Version, as the publisher.
- **D.** Preserve all the copyright notices of the Document.
- **E.** Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- **F.** Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- **G.** Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- **H.** Include an unaltered copy of this License.
- **I.** Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- **J.** Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- **K.** For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- **L.** Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- **M.** Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- **N.** Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- **O.** Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.