A guide to the internals of the GNU linker

Per Bothner, Steve Chamberlain
Cygnus Support

Cygnus Support
Revision: 1.2
TEXinfo 2024-02-10.22

1 The README File

Check the `README` file; it often has useful information that does not appear anywhere else in the directory.

2 How linker emulations are generated

The linker is controlled by linker scripts written in a linker control language. A linker emulation gives the personality of the linker, and is mainly defined by certain builtin scripts. If you want to understand how these builtin scripts are generated, the main file to look at is the `genscripts.sh` shell script, which invoked by the `Makefile` for each "emulation" to generate a set of 5 linker scripts.

For example, for the sun3 emulation used by ld68k, `genscripts.sh` sources the file `sun3.sh`, which sets the emulation parameters, and specifies that the format is a.out, and to use `aout.sc-sh` to generate the linker scripts.

`genscripts.sh` generates 5 different linker scripts, one for each of the `ld` options '`-z`' (default), '`-n`', '`-N`', '`-r`' and '`-Ur`', where each script is slightly different and is generated using the template in `aout.sc-sh` (for the sun3).

3 Porting the linker

Before porting `ld` itself, you will need to port the BFD library; see `../bfd/PORTING`.

The *host* is the system a tool runs *on*. The *target* is the system a tool runs *for*; i.e., a tool can read and write the binaries of the target. Most often, host==target, but `ld` supports cross-linking (and to some extent the same `ld` binary can be used a linker for multiple target architectures).

3.1 Porting to a new host

Pick a name for your host. Call that *host-type*. You need to create the file `config/mh-`*`host-type`*.

3.2 Porting to a new target

Pick a name for your target. Call that *`target`*. You need to create at least `config/mt-`*`target`*. It should contain

```
EMUL=emulation
```

An *emulation* controls the "personality" of `ld`, such as the default linker script. Usually, the *`emulation`* will have the same name as the *`target`*, and you will need to create a new *`emulation`* (see below).

You also need to edit `Makefile.in` and possibly `configure.in`. To see how to do that, search for existing examples (e.g., sun3, sun4, hp300bsd).

3.3 Porting to a new emulation target

Pick a name for your target. Call that *emulation*. Usually, *emulation* and *target* are the same. You need to create at least *emulation*.sh. You will also need to edit Makefile.in. To see how to do that, search for existing examples.

The file *emulation*.sh defines a set of parameters that are used to generate the emulation. Its syntax is that of a Bourne shell script. It is "sourced" by genscripts.sh.

3.4 Writing *emulation*.sh

Usually, *emulation*.sh contains:

```
EMULATION_NAME=emulation
SCRIPT_NAME=script
OUTPUT_FORMAT="target-name"
TEXT_START_ADDR=text-start-addr
PAGE_SIZE=page-size
SEGMENT_SIZE=segment-size  # If different from PAGE_SIZE.
ARCH=arch
```

Here:

target-name
: Matches the filename field of the bfd_target you want to use. (This is a string, and currently the first field.) For an a.out target, *target-name* matches the TARGETNAME defined in ../bfd/*target*.c.

arch
: The architecture: e.g., m68k, sparc, ...

script
: The file *script*.sc-sh is a shell script which, when evaluated (by genscripts.sh), writes a linker script file to standard output. You may need to write a new script. If you use the a.out format or something similar, you can probably set

```
SCRIPT_NAME=aout
```

text-start-addr
page-size
segment-size
: These set the shell variables TEXT_START_ADDR, PAGE_SIZE, and SEGMENT_SIZE for use by *script*.sc-sh. If your script doesn't use these variables, you don't have to define the variables, For emulations using a.out files, you can get these values from ../bfd/*target*.c.

In some cases, you may need more more definitions. For example, if you can't use generic.em, you may need to add:

```
TEMPLATE_NAME=emulation
```

and write your own *emulation*.em file.

3.5 Writing a new linker script *script*.sc-sh

You may need to write a new script file for your emulation.

Your script can use the shell variable `LD_FLAG`, which has the value:

`LD_FLAG=` when building a script to be used by default

`LD_FLAG=n`
: when building a script to be used for '`ld -n`'

`LD_FLAG=N`
: when building a script to be used for '`ld -N`'

`LD_FLAG=r`
: when building a script to be used for '`ld -r`'

`LD_FLAG=u`
: when building a script to be used for '`ld -Ur`'

The variable `RELOCATING` is only set if relocation is happening (i.e., unless the linker is invoked with '`-r`'). Thus your script should has an action *`ACTION`* that should only be done when relocating, express that as:

```
${RELOCATING+ ACTION}
```

In general, this is the case for most assignments, which should look like:

```
${RELOCATING+ _end = .}
```

Also, you should assign absolute addresses to sections only when relocating, so:

```
.text ${RELOCATING+ ${TEXT_START_ADDR}}:
```

The forms:

```
.section { ... } > section
```

should be:

```
.section { ... } > ${RELOCATING+ section}
```

`RELOCATING` is set except when `LD_FLAG=r` or `LD_FLAG=u`. `CONSTRUCTING` is set except when `LD_FLAG=u`.

Alignment of the data segments is controlled by the variables `DATA_ALIGNMENT_` (note trailing underscore), `DATA_ALIGNMENT_n`, `DATA_ALIGNMENT_N`, `DATA_ALIGNMENT_r`, or `DATA_ALIGNMENT_u` depending on the value of `LD_FLAGS`. Normally, the default value works (this is `"ALIGN(${SEGMENT_SIZE})"` for the '`_n`', and '`__`' (default) variants; `"."` for the '`_N`', variant; and `""` for the '`_r`' and '`_u`' variants).

3.6 Handling '`-n`' and '`-N`' style binaries in your linker script

The '`-n`' linker option requests the linker to create a binary with a write-protected text segment, but not demand-pagable (`NMAGIC`). SunOS starts the text segment for demand-paged binaries at 0x2020 and other binaries at 0x2000, since the exec header (0x20 bytes) is paged in with the text. Some other Unix variants do the same.

In that case, the *`emulation`*`.sh` should define:

`NONPAGED_TEXT_START_ADDR`
: The text start address to use when linking with '`-n`' or '`-N`' options.

For example, on a sun4:

```
TEXT_START_ADDR=0x2020
NONPAGED_TEXT_START_ADDR=0x2000
```

The '`-N`' linker option creates a binary with a non-write-protected text segment (`NMAGIC`). This is like '`-n`', except that the data segment needs not be page-aligned.

Table of Contents