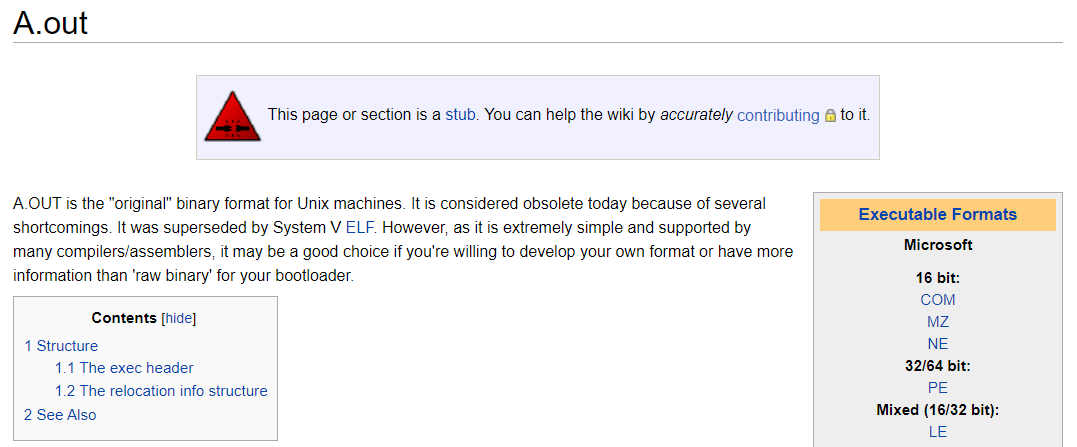
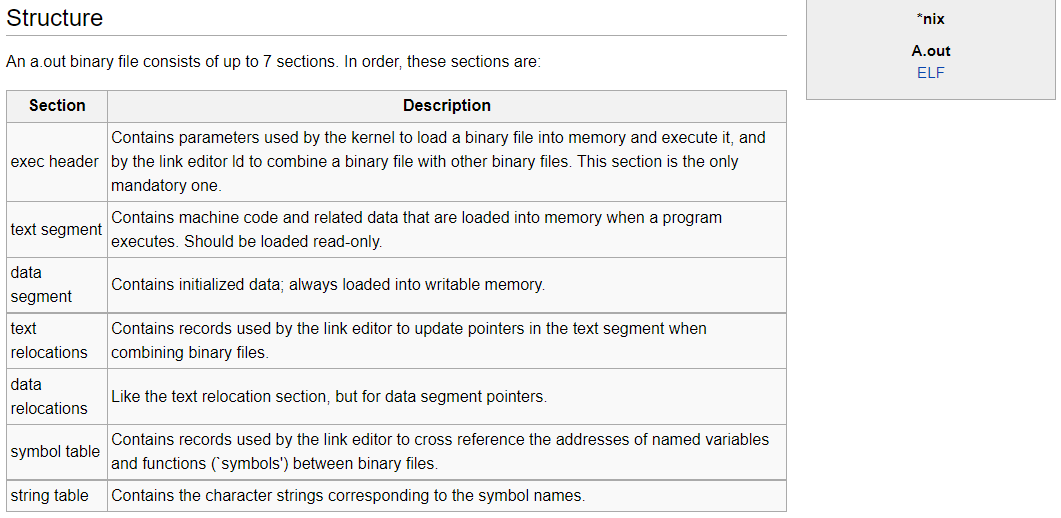
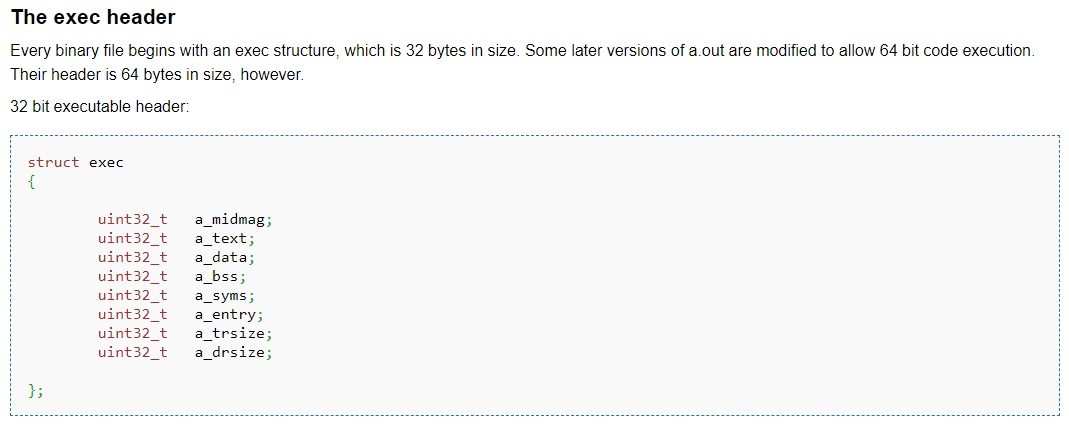
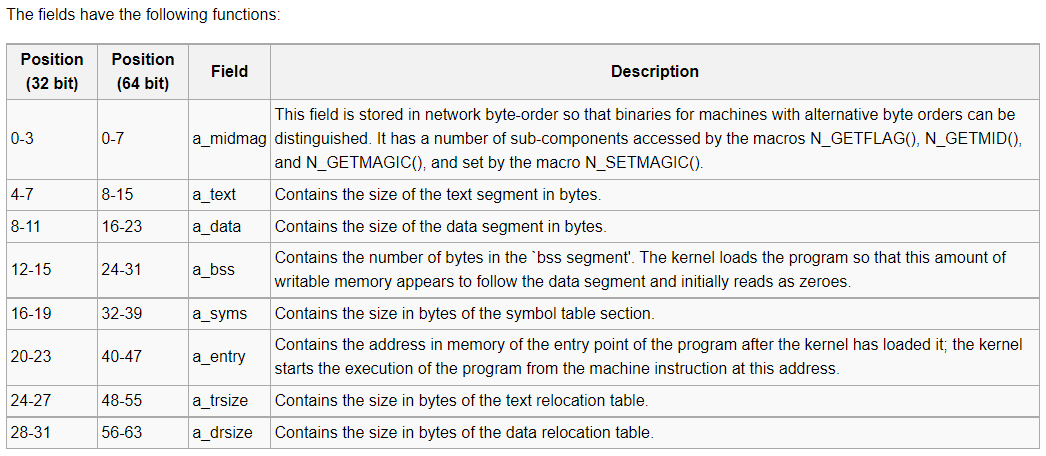
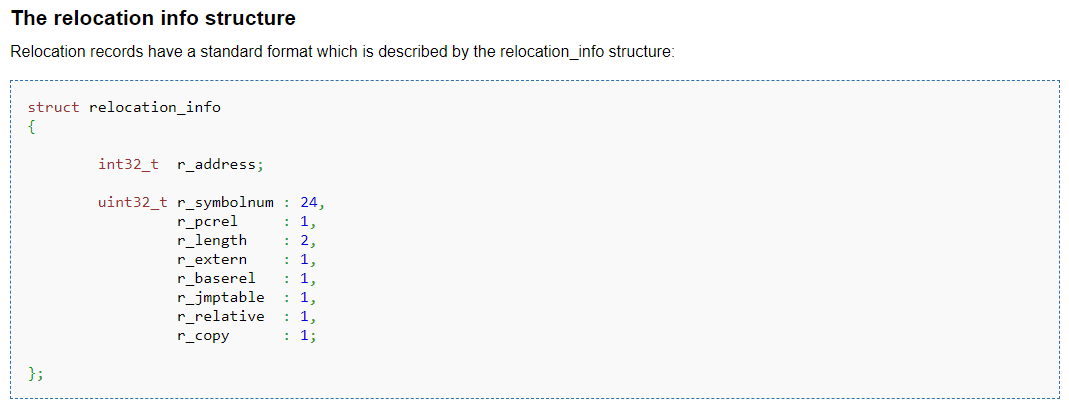
<https://wiki.osdev.org/A.out>

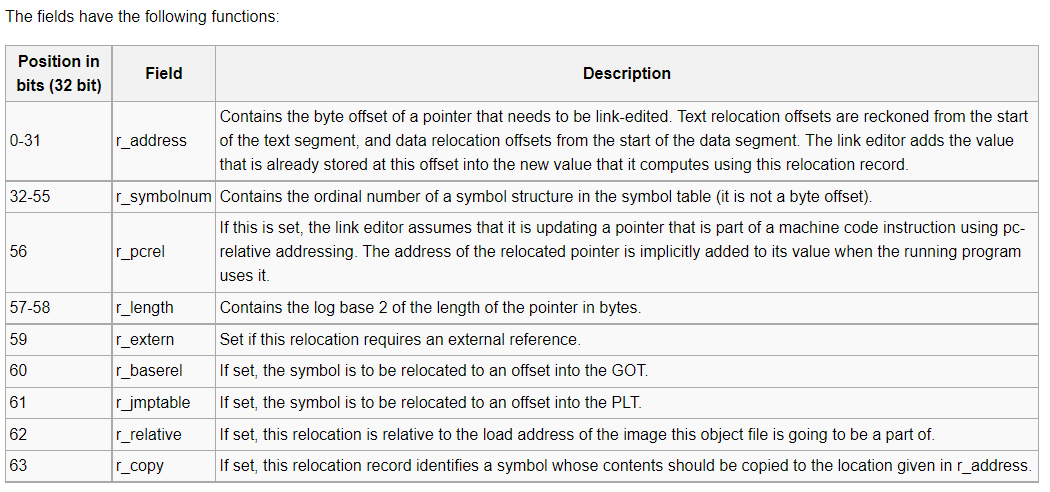


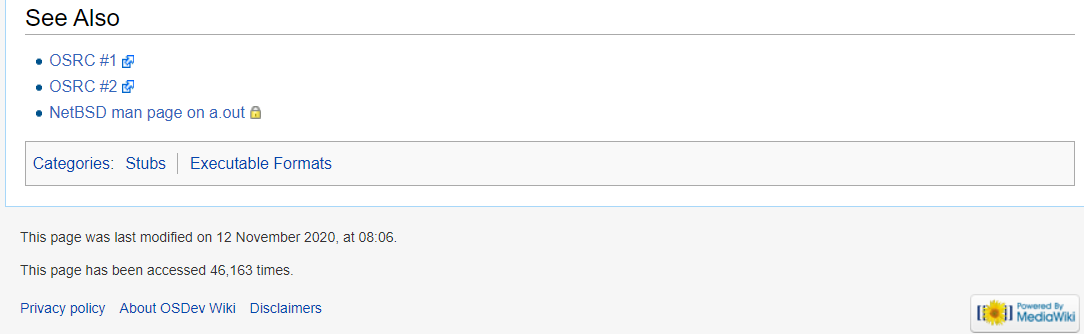












<http://www.linuxmisc.com/16-linux-development/6793e515ea9475b1.htm>

How to compile a.out binaries ?

RM

#1/11

Sat, 12 Nov 2005 21:05:38 GMT

Hi,

I'm trying to run an old Linux kernel that only accept a.out binaries.

I've created a small libc with the kernel headers, but I do not

succeed

in compiling a.out binaries.

I use Slackware 8.1 for compiling, and I test on an old Intel 486.

I have tried to use the "--oformat a.out-i386-linux" with ld

but i get this error:

ld -N -nostdlib ../lib/entry.o test.o ../lib/lib.a -o Bob --oformat

a.out-i386-linux -v

GNU ld version 2.12.90.0.9 20020526

ld: Bob: can not represent section `.rodata' in a.out object file

format

ld: final link failed: Nonrepresentable section on output

The lib.a and others .o have been compiled with gcc -c or as. Is there

some other flag I should turn on in order to make it work?

Thanks,

RMP

Thomas Richte

#2 / 11

Sun, 13 Nov 2005 16:34:16 GMT

Hi,

Have you tried to link your binary statically, i.e. "-static"?

Greetings,

Thomas

RM

#3 / 11

Fri, 18 Nov 2005 14:49:52 GMT

Sorry, but the -static flags does not change anything,

no even more than the -nostdlib.

Is there not anyone that knows how to compile a.out binaires ?

And if not, my only solution is to find old a.out binairies (like sh, ls,

cat, ...) for old Linux (e.g. 0.0x).

Thanks

RMP

trec..

#4 / 11

Fri, 18 Nov 2005 15:12:53 GMT

|Sorry, but the -static flags does not change anything,

|no even more than the -nostdlib.

|

|Is there not anyone that knows how to compile a.out binaires ?

|

|And if not, my only solution is to find old a.out binairies (like sh, ls,

|cat, ...) for old Linux (e.g. 0.0x).

Try the -oformat option to ld, which can be passed by -Xlinker option on

gcc.

RM

#5 / 11

Fri, 18 Nov 2005 16:48:15 GMT

Well i think we've misunderstood, so let me explain my problem again.

I got a source code in C and Intel x86 assembly (AT&T syntax).

I do NOT use any shared libs. A part of this code is my own libc.

I have compiled the lib files with gcc -c and put them in lib/lib.a

with ar.

I use as to assemble and obtain entry.o, an entry point with \_start symbol.

It calls main() and then use \_exit() syscall with the main() return value.

I've created a simple C source to test my lib (using printf(), getpid(), etc..)

and I've linked all this using ld :

ld -nostdlib ../lib/entry.o test.o ../lib/lib.a -o Test

I obtain an ELF binary that works fine without any shared libs.

But the final goal was to create a.out binaires, that do not use shared libs,

in order to use them on an old Linux (0.01) i've compiled on my 486.

The kernel works fine, the MINIX root partition too.

The problem is that i need to use a.out binaries to test this kernel. Before

trying to code a small shell, i want to try simple "HelloWorld" binaries.

So i tried to use the "--oformat" option in ld, but it fails:

ld ../lib/entry.o test.o ../lib/lib.a --oformat a.out-i386-linux -nostdlib

ld: a.out: can not represent section `.rodata' in a.out object file format

ld: final link failed: Nonrepresentable section on output

I've also tried to rename section .rodata to .date with objcopy, but it

fails too:

objcopy Test -O a.out-i386-linux test --rename-section .rodata=.data

BFD: test: can not represent section `.data' in a.out object file format

objcopy: test: Nonrepresentable section on output

So now that i (shortly ;) ) have explained my problem, my question is:

How can i compile a.out binairies with GCC (or ld, which is the same) ?

Thanks,

RMP

iz..

#6 / 11

Fri, 18 Nov 2005 16:52:22 GMT

|So i tried to use the "--oformat" option in ld, but it fails:

|

|ld ../lib/entry.o test.o ../lib/lib.a --oformat a.out-i386-linux -nostdlib

|ld: a.out: can not represent section `.rodata' in a.out object file format

|ld: final link failed: Nonrepresentable section on output

It looks like the new gcc and gas generate this section and you may have

to get an old version of the compiler tools. Maybe you can do something

by generating asm and then editing it by hand to rename the section.

RM

#7 / 11

Sat, 19 Nov 2005 15:53:02 GMT

I'm trying to compile old GCCs (ftp.gnu.org/old-gnu/gcc), but it seems

that

there's compatibilty problem with my libc (which is recent, of

course). Does anyone knows a version that can be compiled on a modern

box ?

Or some way to get pre-compiled old binaries, or to directly get

MINIX version (which are a.out) of ls, sh, etc...

Or is there a way to make GCC compile for this fuc\*&! format (or a way

to

use objcopy) ?

RMP

rz..

#8 / 11

Sat, 19 Nov 2005 16:37:24 GMT

|Or some way to get pre-compiled old binaries, or to directly get

|MINIX version (which are a.out) of ls, sh, etc...

There are some old distros there which use libc and a.out, I think

TinyLinux may be one. Good luck.

Waldek Hebis

#9 / 11

Wed, 23 Nov 2005 09:01:56 GMT

: Well i think we've misunderstood, so let me explain my problem again.

: I got a source code in C and Intel x86 assembly (AT&T syntax).

: I do NOT use any shared libs. A part of this code is my own libc.

: I have compiled the lib files with gcc -c and put them in lib/lib.a

: with ar.

: I use as to assemble and obtain entry.o, an entry point with \_start symbol.

: It calls main() and then use \_exit() syscall with the main() return value.

: I've created a simple C source to test my lib (using printf(), getpid(), etc..)

: and I've linked all this using ld :

: ld -nostdlib ../lib/entry.o test.o ../lib/lib.a -o Test

: I obtain an ELF binary that works fine without any shared libs.

: But the final goal was to create a.out binaires, that do not use shared libs,

: in order to use them on an old Linux (0.01) i've compiled on my 486.

: The kernel works fine, the MINIX root partition too.

: The problem is that i need to use a.out binaries to test this kernel. Before

: trying to code a small shell, i want to try simple "HelloWorld" binaries.

: So i tried to use the "--oformat" option in ld, but it fails:

: ld ../lib/entry.o test.o ../lib/lib.a --oformat a.out-i386-linux -nostdlib

: ld: a.out: can not represent section `.rodata' in a.out object file format

: ld: final link failed: Nonrepresentable section on output

: I've also tried to rename section .rodata to .date with objcopy, but it

: fails too:

: objcopy Test -O a.out-i386-linux test --rename-section .rodata=.data

: BFD: test: can not represent section `.data' in a.out object file format

: objcopy: test: Nonrepresentable section on output

: So now that i (shortly ;) ) have explained my problem, my question is:

: How can i compile a.out binairies with GCC (or ld, which is the same) ?

objcopy --rename-section .rodata=.data test.o test1.o

objcopy --rename-section .rodata=.data ../lib/entry.o entry1.o

... as needed for all object files

ld --oformat a.out-i386-linux entry1.o test1.o -o test1

You may also change .rodata to .text (in a.out format read only

data is stored together with machine code, changing to .data

makes it read-write).

--

Waldek Hebisch

hebi...@math.uni.wroc.pl or hebi...@hera.math.uni.wroc.pl

RM

#10 / 11

Sat, 26 Nov 2005 16:39:30 GMT

Well, at first it thought it works. But then i realize that it was a

QMAGIC

a.out binary. I want to create ZMAGIC, which seems to be possible with

objcopy.

So i did this:

ld -s -x -nostdlib entry1.o test1.o lib/lib1.a -o Test

objcopy -O a.out-i386-linux -R .note -R .comment Test Test-aout

It seems to works but produce a buggy ZMAGIC binary. Just look at my

objdump result:

# The ELF version

$ objdump -x Test

Test: file format elf32-i386

Test

architecture: i386, flags 0x00000102:

EXEC\_P, D\_PAGED

start address 0x08048080

Program Header:

LOAD off 0x00000000 vaddr 0x08048000 paddr 0x08048000 align

2\*\*12

filesz 0x00000673 memsz 0x00000673 flags r-x

LOAD off 0x00000680 vaddr 0x08049680 paddr 0x08049680 align

2\*\*12

filesz 0x00000190 memsz 0x000005a4 flags rw-

Sections:

Idx Name Size VMA LMA File off Algn

0 .text 000005f3 08048080 08048080 00000080 2\*\*4

CONTENTS, ALLOC, LOAD, READONLY, CODE

1 .data 00000190 08049680 08049680 00000680 2\*\*5

CONTENTS, ALLOC, LOAD, DATA

2 .bss 00000404 08049820 08049820 00000820 2\*\*5

ALLOC

objdump: Test: no symbols

# Now the ZMAGIC version

$ objdump -x Test-aout

Test-aout: file format a.out-i386-linux

Test-aout

architecture: i386, flags 0x00000182:

EXEC\_P, WP\_TEXT, D\_PAGED

start address 0x0000000008048080

Sections:

Idx Name Size VMA LMA File

off Algn

0 .text 000015e0 0000000000000020 0000000000000020

00000020 2\*\*3

CONTENTS, ALLOC, LOAD, CODE

1 .data 00001000 0000000000002000 0000000000002000

00001600 2\*\*3

CONTENTS, ALLOC, LOAD, DATA

2 .bss 00000000 0000000000003000 0000000000003000

00000000 2\*\*3

ALLOC

objdump: Test-aout: no symbols

# end of dump

As you can see, size has increased (strongly), and when i try to dump,

I see that sections .data and .text sizes have been extended with

zero-filling (the originals code and text are still here).

Last thing, i created a small program that read an a.out file, reads

the header and performs the same checks that the kernel does. All are

ok but one:

when you add sections size and header size, you find something greater

than real

file size (about 3696 bytes more).

I start to think that it is not possible to make ZMAGIC a.out binaries

now, and that my project will be only a working kernel with no

binaries to test it.

RMP

Waldek Hebis

#11 / 11

Sun, 27 Nov 2005 08:00:58 GMT

: > objcopy --rename-section .rodata=.data test.o test1.o

: > objcopy --rename-section .rodata=.data ../lib/entry.o entry1.o

: >

: > ... as needed for all object files

: >

: > ld --oformat a.out-i386-linux entry1.o test1.o -o test1

: >

: > You may also change .rodata to .text (in a.out format read only

: > data is stored together with machine code, changing to .data

: > makes it read-write).

: Well, at first it thought it works. But then i realize that it was a

: QMAGIC

: a.out binary. I want to create ZMAGIC, which seems to be possible with

: objcopy.

: So i did this:

: ld -s -x -nostdlib entry1.o test1.o lib/lib1.a -o Test

: objcopy -O a.out-i386-linux -R .note -R .comment Test Test-aout

: It seems to works but produce a buggy ZMAGIC binary. Just look at my

: objdump result:

<snip>

: As you can see, size has increased (strongly), and when i try to dump,

: I see that sections .data and .text sizes have been extended with

: zero-filling (the originals code and text are still here).

: Last thing, i created a small program that read an a.out file, reads

: the header and performs the same checks that the kernel does. All are

: ok but one:

: when you add sections size and header size, you find something greater

: than real

: file size (about 3696 bytes more).

: I start to think that it is not possible to make ZMAGIC a.out binaries

: now, and that my project will be only a working kernel with no

: binaries to test it.

As long as I can see it is not possible to run ZMAGIC binary on

modern kernel, so no wonder that it is hard to make a correct one.

However, a.out is a very simple format, so you should be able to

correct problems by hand (edit header with a hex editor and pad

with zeroes at the end). ZMAGIC executable has a header (see

/usr/include/linux/a.out.h) padded to 1024 butes, then code

beeing multiple of page size, then data again in multiple

of page size. Appearently objcopy is not padding the executable.

If you want to automate making ZMAGIC binares probably the

best way is first convert to binary (the way the kernel is

made) and then write a little program to prepend correct

header.

--

Waldek Hebisch

hebi...@math.uni.wroc.pl or hebi...@hera.math.uni.wroc.pl

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

https://www.reddit.com/r/linuxquestions/comments/ebosal/how\_to\_make\_gccld\_output\_an\_executable\_with\_aout/