

[PATCH for -rc] kbuild: fix sh64 section mismatch problems

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-06/msg04407.html>

- *From:* Sam Ravnborg <sam@xxxxxxxxxxxxx>
 - *Date:* Mon, 11 Jun 2007 21:55:52 +0200
-

Hi Linus.

Please apply following 2 liners fix.
It will fix a lot of false section mismatch warnings on sh64 and
Paul asked to have in included before the relase hit the street.

Please pull this single patch from:
<git://master.kernel.org/pub/scm/linux/kernel/git/sam/kbuild-fix.git>

Sam

Patch below for reference:

modpost.c | 3 ++-
1 file changed, 2 insertions(+), 1 deletion(-)

From 2648a53acf16a837e11836203aadb219bd951a1e Mon Sep 17 00:00:00 2001

From: Sam Ravnborg <sam@xxxxxxxxxxxxx>
Date: Mon, 11 Jun 2007 21:52:04 +0200
Subject: [PATCH] kbuild: fix sh64 section mismatch problems

There's a special .cranges section that is almost always generated,
with data being moved to the appropriate section by the linker at a later
stage.

To give a bit of background, sh64 has both a native SHmedia instruction
set (32-bit instructions) and SHcompact (which is compatability with
normal SH -- 16-bit, a massively reduced register set, etc.). code ranges
are emitted when we're using the 32-bit ABI, but not the 64-bit one.

It is a special staging section used solely by binutils where code with
different flags get placed (more specifically differing flags for input
and output sections), before being lazily merged by the linker.

The closest I've been able to find to documentation is:
<http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/ld/emultempl/sh64elf.em?rev=1.10&content-type=text/x-cvsweb-m>

[PATCH for -rc] kbuild: fix sh64 section mismatch problems

It's an array of 8-byte Elf32_CRange structure given in

<http://sources.redhat.com/cgi-bin/cvsweb.cgi/src/bfd/elf32-sh64.h?rev=1.4&content-type=text/x-cvsweb-markup&c>
that describes for which ISA a range is used.

Silence the warnings by allowing references from .init.text to .cranges.

The following warnings are fixed:

WARNING: init/built-in.o(.cranges+0x0): Section mismatch: reference to .init.text:
WARNING: init/built-in.o(.cranges+0xa): Section mismatch: reference to .init.text:
WARNING: init/built-in.o(.cranges+0x14): Section mismatch: reference to .init.text:
WARNING: init/built-in.o(.cranges+0x1e): Section mismatch: reference to .init.text:
WARNING: init/built-in.o(.cranges+0x28): Section mismatch: reference to .init.text:
WARNING: init/built-in.o(.cranges+0x32): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x50): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x5a): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x64): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0xfa): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x104): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x10e): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x14a): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x154): Section mismatch: reference to .init.text:
WARNING: kernel/built-in.o(.cranges+0x15e): Section mismatch: reference to .init.text:
WARNING: mm/built-in.o(.cranges+0x6e): Section mismatch: reference to .init.text:
WARNING: mm/built-in.o(.cranges+0x78): Section mismatch: reference to .init.text:
WARNING: mm/built-in.o(.cranges+0x82): Section mismatch: reference to .init.text:
WARNING: mm/built-in.o(.cranges+0xaa): Section mismatch: reference to .init.text:
WARNING: fs/built-in.o(.cranges+0x136): Section mismatch: reference to .init.text:
WARNING: fs/built-in.o(.cranges+0x140): Section mismatch: reference to .init.text:
WARNING: fs/built-in.o(.cranges+0x14a): Section mismatch: reference to .init.text:
WARNING: fs/built-in.o(.cranges+0x168): Section mismatch: reference to .init.text:
WARNING: fs/built-in.o(.cranges+0x1f4): Section mismatch: reference to .init.text:
WARNING: fs/built-in.o(.cranges+0x1fe): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x302): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x30c): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x316): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x3a2): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x3ac): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x4ce): Section mismatch: reference to .init.text:
WARNING: net/built-in.o(.cranges+0x4d8): Section mismatch: reference to .init.text:

Signed-off-by: Paul Mundt <lethal@xxxxxxxxxxxx>

Cc: Kaz Kojima <kkojima@xxxxxxxxxxxx>

Signed-off-by: Sam Ravnborg <sam@xxxxxxxxxxxx>

scripts/mod/modpost.c | 2 ++

1 files changed, 2 insertions(+), 0 deletions(-)

diff --git a/scripts/mod/modpost.c b/scripts/mod/modpost.c

index 8e5610d..3645e98 100644

[PATCH for -rc] kbuild: fix sh64 section mismatch problems

[PATCH for -rc] kbuild: fix sh64 section mismatch problems

```
--- a/scripts/mod/modpost.c
+++ b/scripts/mod/modpost.c
@@ -1052,6 +1052,7 @@ static int init_section_ref_ok(const char *name)
".plt", /* seen on ARCH=um build on x86_64. Harmless */
"__ftr_fixup", /* powerpc cpu feature fixup */
"__fw_ftr_fixup", /* powerpc firmware feature fixup */
+ ".cranges", /* used by sh64 */
NULL
};
/* Start of section names */
@@ -1132,6 +1133,7 @@ static int exit_section_ref_ok(const char *name)
".fixup",
".smp_locks",
".plt", /* seen on ARCH=um build on x86_64. Harmless */
+ ".cranges", /* used by sh64 */
NULL
};
/* Start of section names */
--
1.5.1.rc3.1544.g8a923
```

—
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>