

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-09/msg09117.html>

- *From:* Mathieu Desnoyers <compudj@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 30 Sep 2006 14:04:43 -0400
-

Hi,

New release : compile fix when markers disabled.

Mathieu

---BEGIN---

```
diff --git a/Makefile b/Makefile
index 1700d3f..78ed30f 100644
--- a/Makefile
+++ b/Makefile
@@ -301,7 +301,8 @@ # Use LINUXINCLUDE when you must referen
# Needed to be compatible with the O= option
LINUXINCLUDE := -Iinclude \
$(if $(KBUILD_SRC),-Iinclude2 -I$(srctree)/include) \
--include include/linux/autoconf.h
+ -include include/linux/autoconf.h \
+ -include include/linux/marker.h
```

```
CPPFLAGS := -D__KERNEL__ $(LINUXINCLUDE)
```

```
diff --git a/arch/alpha/Kconfig b/arch/alpha/Kconfig
index 213c785..95c1d1b 100644
--- a/arch/alpha/Kconfig
+++ b/arch/alpha/Kconfig
@@ -630,6 +630,12 @@ source "fs/Kconfig"
```

```
source "arch/alpha/oprofile/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/alpha/Kconfig.debug"
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
source "security/Kconfig"
diff --git a/arch/arm/Kconfig b/arch/arm/Kconfig
index 08b7cc9..4c03f09 100644
--- a/arch/arm/Kconfig
+++ b/arch/arm/Kconfig
@@ -871,6 +871,12 @@ source "fs/Kconfig"

source "arch/arm/oprofile/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/arm/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/arm26/Kconfig b/arch/arm26/Kconfig
index cf4ebf4..f34e157 100644
--- a/arch/arm26/Kconfig
+++ b/arch/arm26/Kconfig
@@ -232,6 +232,12 @@ source "drivers/misc/Kconfig"

source "drivers/usb/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/arm26/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/cris/Kconfig b/arch/cris/Kconfig
index 856b665..84f8efd 100644
--- a/arch/cris/Kconfig
+++ b/arch/cris/Kconfig
@@ -179,6 +179,12 @@ source "sound/Kconfig"

source "drivers/usb/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/cris/Kconfig.debug"
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
source "security/Kconfig"
diff --git a/arch/frv/Kconfig b/arch/frv/Kconfig
index 95a3892..8708a75 100644
--- a/arch/frv/Kconfig
+++ b/arch/frv/Kconfig
@@ -345,6 +345,12 @@ source "drivers/Kconfig"
```

```
source "fs/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/frv/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/h8300/Kconfig b/arch/h8300/Kconfig
index cabf0bf..bdb67ed 100644
--- a/arch/h8300/Kconfig
+++ b/arch/h8300/Kconfig
@@ -197,6 +197,12 @@ endmenu
```

```
source "fs/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/h8300/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/i386/Kconfig b/arch/i386/Kconfig
index 8dfa305..e6cc7da 100644
--- a/arch/i386/Kconfig
+++ b/arch/i386/Kconfig
@@ -1081,6 +1081,9 @@ config KPROBES
a probepoint and specifies the callback. Kprobes is useful
for kernel debugging, non-intrusive instrumentation and testing.
If in doubt, say "N".
```

```
+
+source "kernel/Kconfig.marker"
+
endmenu
```

```
source "arch/i386/Kconfig.debug"
diff --git a/arch/ia64/Kconfig b/arch/ia64/Kconfig
index 0f3076a..2342975 100644
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
--- a/arch/ia64/Kconfig
+++ b/arch/ia64/Kconfig
@@ -499,6 +499,9 @@ config KPROBES
a probepoint and specifies the callback. Kprobes is useful
for kernel debugging, non-intrusive instrumentation and testing.
If in doubt, say "N".
+
+source "kernel/Kconfig.marker"
+
endmenu

source "arch/ia64/Kconfig.debug"
diff --git a/arch/m32r/Kconfig b/arch/m32r/Kconfig
index 41fd490..50f0a8e 100644
--- a/arch/m32r/Kconfig
+++ b/arch/m32r/Kconfig
@@ -386,6 +386,12 @@ source "fs/Kconfig"

source "arch/m32r/oprofile/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/m32r/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/m68k/Kconfig b/arch/m68k/Kconfig
index 805b81f..5af9e00 100644
--- a/arch/m68k/Kconfig
+++ b/arch/m68k/Kconfig
@@ -652,6 +652,12 @@ endmenu

source "fs/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/m68k/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/m68knommu/Kconfig b/arch/m68knommu/Kconfig
index 3cde682..bd01cc0 100644
--- a/arch/m68knommu/Kconfig
+++ b/arch/m68knommu/Kconfig
@@ -652,6 +652,12 @@ source "drivers/Kconfig"
```

```
source "fs/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/m68knommu/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/mips/Kconfig b/arch/mips/Kconfig
index e8ff09f..4d5dbf9 100644
--- a/arch/mips/Kconfig
+++ b/arch/mips/Kconfig
@@ -1850,6 +1850,12 @@ source "fs/Kconfig"

source "arch/mips/oprofile/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/mips/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/parisc/Kconfig b/arch/parisc/Kconfig
index 910fb3a..a0a7dd7 100644
--- a/arch/parisc/Kconfig
+++ b/arch/parisc/Kconfig
@@ -251,6 +251,12 @@ source "fs/Kconfig"

source "arch/parisc/oprofile/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/parisc/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/powerpc/Kconfig b/arch/powerpc/Kconfig
index 6729c98..44f25b5 100644
--- a/arch/powerpc/Kconfig
+++ b/arch/powerpc/Kconfig
@@ -1011,6 +1011,9 @@ config KPROBES
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

a probepoint and specifies the callback. Kprobes is useful for kernel debugging, non-intrusive instrumentation and testing. If in doubt, say "N".

```
+
+source "kernel/Kconfig.marker"
+
endmenu

source "arch/powerpc/Kconfig.debug"
diff --git a/arch/ppc/Kconfig b/arch/ppc/Kconfig
index e9a8f5d..ca871ab 100644
--- a/arch/ppc/Kconfig
+++ b/arch/ppc/Kconfig
@@ -1412,6 +1412,12 @@ source "lib/Kconfig"
```

```
source "arch/powerpc/oprofile/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/ppc/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/s390/Kconfig b/arch/s390/Kconfig
index 01c5c08..57600b5 100644
--- a/arch/s390/Kconfig
+++ b/arch/s390/Kconfig
@@ -476,6 +476,12 @@ source "fs/Kconfig"
```

```
source "arch/s390/oprofile/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/s390/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/sh/Kconfig b/arch/sh/Kconfig
index 2bcecf4..0fd24d3 100644
--- a/arch/sh/Kconfig
+++ b/arch/sh/Kconfig
@@ -644,6 +644,12 @@ source "fs/Kconfig"
```

```
source "arch/sh/oprofile/Kconfig"
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/sh/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/sh64/Kconfig b/arch/sh64/Kconfig
index 58c678e..f4ecb4d 100644
--- a/arch/sh64/Kconfig
+++ b/arch/sh64/Kconfig
@@ -276,6 +276,12 @@ source "fs/Kconfig"
```

```
source "arch/sh64/oprofile/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/sh64/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/sparc/Kconfig b/arch/sparc/Kconfig
index 9431e96..de02311 100644
--- a/arch/sparc/Kconfig
+++ b/arch/sparc/Kconfig
@@ -289,6 +289,12 @@ endmenu
```

```
source "fs/Kconfig"
```

```
+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/sparc/Kconfig.debug"
```

```
source "security/Kconfig"
diff --git a/arch/sparc64/Kconfig b/arch/sparc64/Kconfig
index 43a66f5..971f5e2 100644
--- a/arch/sparc64/Kconfig
+++ b/arch/sparc64/Kconfig
@@ -419,6 +419,9 @@ config KPROBES
a probepoint and specifies the callback. Kprobes is useful
for kernel debugging, non-intrusive instrumentation and testing.
If in doubt, say "N".
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+
+source "kernel/Kconfig.marker"
+
endmenu

source "arch/sparc64/Kconfig.debug"
diff --git a/arch/um/Kconfig b/arch/um/Kconfig
index 76e85bb..9481883 100644
--- a/arch/um/Kconfig
+++ b/arch/um/Kconfig
@@ -312,4 +312,10 @@ config INPUT
bool
default n

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/um/Kconfig.debug"
diff --git a/arch/v850/Kconfig b/arch/v850/Kconfig
index 37ec644..6ce651f 100644
--- a/arch/v850/Kconfig
+++ b/arch/v850/Kconfig
@@ -324,6 +324,12 @@ source "sound/Kconfig"

source "drivers/usb/Kconfig"

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/v850/Kconfig.debug"

source "security/Kconfig"
diff --git a/arch/x86_64/Kconfig b/arch/x86_64/Kconfig
index 408d44a..6feb011 100644
--- a/arch/x86_64/Kconfig
+++ b/arch/x86_64/Kconfig
@@ -611,6 +611,9 @@ config KPROBES
a probepoint and specifies the callback. Kprobes is useful
for kernel debugging, non-intrusive instrumentation and testing.
If in doubt, say "N".
+
+source "kernel/Kconfig.marker"
+
endmenu
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
source "arch/x86_64/Kconfig.debug"
diff --git a/arch/xtensa/Kconfig b/arch/xtensa/Kconfig
index dbeb350..413abb8 100644
--- a/arch/xtensa/Kconfig
+++ b/arch/xtensa/Kconfig
@@ -255,6 +255,12 @@ config EMBEDDED_RAMDISK_IMAGE
provide one yourself.
endmenu

+menu "Instrumentation Support"
+
+source "kernel/Kconfig.marker"
+
+endmenu
+
source "arch/xtensa/Kconfig.debug"

source "security/Kconfig"
diff --git a/include/asm-alpha/marker.h b/include/asm-alpha/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-alpha/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-arm/marker.h b/include/asm-arm/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-arm/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ */
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-arm26/marker.h b/include/asm-arm26/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-arm26/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-cris/marker.h b/include/asm-cris/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-cris/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-frv/marker.h b/include/asm-frv/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-frv/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-generic/marker.h b/include/asm-generic/marker.h
new file mode 100644
index 0000000..aaa0338
--- /dev/null
+++ b/include/asm-generic/marker.h
@@ -0,0 +1,44 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Generic header.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+struct __mark_marker_c {
+ const char *name;
+ marker_probe_func **call;
+ const char *format;
+} __attribute__((packed));
+
+struct __mark_marker {
+ const struct __mark_marker_c *cmark;
+ volatile char *enable;
+} __attribute__((packed));
+
+#ifdef CONFIG_MARKERS
+
+#define MARK(name, format, args...) \
+ do { \
+ static marker_probe_func *__mark_call_##name = \
+ __mark_empty_function; \
+ volatile static char __marker_enable_##name = 0; \
+ static const struct __mark_marker_c __mark_c_##name \
+ __attribute__((section(".markers.c"))) = \
+ { #name, &__mark_call_##name, format } ; \
+ static const struct __mark_marker __mark_##name \
+ __attribute__((section(".markers"), unused)) = \
+ { &__mark_c_##name, &__marker_enable_##name } ; \
+ if (unlikely(__marker_enable_##name)) { \
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ preempt_disable(); \
+ (*__mark_call_###name)(format, ## args); \
+ preempt_enable_no_resched(); \
+ } \
+ } while(0)
+
+
+#define MARK_ENABLE_IMMEDIATE_OFFSET 0
+
+#endif
diff --git a/include/asm-generic/vmlinux.lds.h b/include/asm-generic/vmlinux.lds.h
index 9d11550..d029043 100644
--- a/include/asm-generic/vmlinux.lds.h
+++ b/include/asm-generic/vmlinux.lds.h
@@ -90,6 +90,12 @@ #define RODATA \
__ksymtab_strings : AT(ADDR(__ksymtab_strings) - LOAD_OFFSET) { \
*(__ksymtab_strings) \
} \
+ /* Kernel markers : pointers */ \
+ .markers : AT(ADDR(.markers) - LOAD_OFFSET) { \
+ VMLINUX_SYMBOL(__start__markers) = .; \
+ *(.markers) \
+ VMLINUX_SYMBOL(__stop__markers) = .; \
+ } \
__end_rodatabytes = .; \
. = ALIGN(4096); \
\
diff --git a/include/asm-h8300/marker.h b/include/asm-h8300/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-h8300/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-i386/marker.h b/include/asm-i386/marker.h
new file mode 100644
index 0000000..ed4bc05
--- /dev/null
+++ b/include/asm-i386/marker.h
```

```

@@ -0,0 +1,53 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. i386 architecture optimisations.
+ *
+ * (C) Copyright 2006 Mathieu Desnoyers <mathieu.desnoyers@xxxxxxxxxxx>
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+
+struct __mark_marker_c {
+ const char *name;
+ marker_probe_func **call;
+ const char *format;
+} __attribute__((packed));
+
+struct __mark_marker {
+ struct __mark_marker_c *cmark;
+ volatile char *enable;
+} __attribute__((packed));
+
+#ifdef CONFIG_MARKERS
+#define MARK(name, format, args...) \
+ do { \
+ static marker_probe_func *__mark_call_###name = \
+ __mark_empty_function; \
+ static const struct __mark_marker_c __mark_c_###name \
+ __attribute__((section(".markers.c"))) = \
+ { #name, &__mark_call_###name, format } ; \
+ char condition; \
+ asm volatile( ".section .markers, \"a\";\n\t" \
+ ".long %1, 0f;\n\t" \
+ ".previous;\n\t" \
+ ".align 2\n\t" \
+ "0:\n\t" \
+ "movb $0,%0;\n\t" \
+ : "=r" (condition) \
+ : "m" (__mark_c_###name)); \
+ __mark_check_format(format, ## args); \
+ if (unlikely(condition)) { \
+ preempt_disable(); \
+ (*__mark_call_###name)(format, ## args); \
+ preempt_enable_no_resched(); \
+ } \
+ } while(0)
+
+/* Offset of the immediate value from the start of the movb instruction, in
+ * bytes. */

```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ #define MARK_ENABLE_IMMEDIATE_OFFSET 1
+
+ #endif
diff --git a/include/asm-ia64/marker.h b/include/asm-ia64/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-ia64/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+ #include <asm-generic/marker.h>
diff --git a/include/asm-m32r/marker.h b/include/asm-m32r/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-m32r/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+ #include <asm-generic/marker.h>
diff --git a/include/asm-m68k/marker.h b/include/asm-m68k/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-m68k/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-m68knommu/marker.h b/include/asm-m68knommu/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-m68knommu/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-mips/marker.h b/include/asm-mips/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-mips/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-parisc/marker.h b/include/asm-parisc/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+++ b/include/asm-parisc/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-powerpc/marker.h b/include/asm-powerpc/marker.h
new file mode 100644
index 0000000..25fbd03
--- /dev/null
+++ b/include/asm-powerpc/marker.h
@@ -0,0 +1,57 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. PowerPC architecture
+ * optimisations.
+ *
+ * (C) Copyright 2006 Mathieu Desnoyers <mathieu.desnoyers@xxxxxxxxxxx>
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm/asm-compat.h>
+
+struct __mark_marker_c {
+ const char *name;
+ marker_probe_func **call;
+ const char *format;
+} __attribute__((packed));
+
+struct __mark_marker {
+ struct __mark_marker_c *cmark;
+ volatile short *enable;
+} __attribute__((packed));
+
+#ifdef CONFIG_MARKERS
+
+#define MARK(name, format, args...) \
+ do { \
+ static marker_probe_func *__mark_call_##name = \
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ __mark_empty_function; \
+ static const struct __mark_marker_c __mark_c_###name \
+ __attribute__((section(".markers.c"), unused)) = \
+ { #name, &__mark_call_###name, format } ; \
+ char condition; \
+ asm volatile( ".section .markers, \"a\";\n\t" \
+ PPC_LONG "%1, 0f;\n\t" \
+ ".previous;\n\t" \
+ ".align 4\n\t" \
+ "0:\n\t" \
+ "li %0,0;\n\t" \
+ : "=r" (condition) \
+ : "i" (&__mark_c_###name)); \
+ __mark_check_format(format, ## args); \
+ if (unlikely(condition)) { \
+ preempt_disable(); \
+ (*__mark_call_###name)(format, ## args); \
+ preempt_enable_no_resched(); \
+ } \
+ } while(0)
+
+
+/* Offset of the immediate value from the start of the addi instruction (result
+ * of the li mnemonic), in bytes. */
+#define MARK_ENABLE_IMMEDIATE_OFFSET 2
+
+#endif
diff --git a/include/asm-ppc/marker.h b/include/asm-ppc/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-ppc/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-ppc64/marker.h b/include/asm-ppc64/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-ppc64/marker.h
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-s390/marker.h b/include/asm-s390/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-s390/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-sh/marker.h b/include/asm-sh/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-sh/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+#include <asm-generic/marker.h>
diff --git a/include/asm-sh64/marker.h b/include/asm-sh64/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-sh64/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-sparc/marker.h b/include/asm-sparc/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-sparc/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-sparc64/marker.h b/include/asm-sparc64/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-sparc64/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-um/marker.h b/include/asm-um/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-um/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-v850/marker.h b/include/asm-v850/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-v850/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-x86_64/marker.h b/include/asm-x86_64/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-x86_64/marker.h
@@ -0,0 +1,13 @@
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/asm-xtensa/marker.h b/include/asm-xtensa/marker.h
new file mode 100644
index 0000000..8d9467f
--- /dev/null
+++ b/include/asm-xtensa/marker.h
@@ -0,0 +1,13 @@
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing. Architecture specific
+ * optimisations.
+ *
+ * No optimisation implemented.
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#include <asm-generic/marker.h>
diff --git a/include/linux/marker.h b/include/linux/marker.h
new file mode 100644
index 0000000..71909f0
--- /dev/null
+++ b/include/linux/marker.h
@@ -0,0 +1,65 @@
+#ifndef _LINUX_MARKER_H
+#define _LINUX_MARKER_H
+
+/*
+ * marker.h
+ *
+ * Code markup for dynamic and static tracing.
+ *
+ * Example :
+ * MARK(subsystem_event, "%d %s", someint, somestring);
+ * Where :
+ * - Subsystem is the name of your subsystem.
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ * – event is the name of the event to mark.
+ * – "%d %s" is the formatted string for printk.
+ * – someint is an integer.
+ * – somestring is a char *.
+ *
+ * – Dynamically overridable function call based on marker mechanism
+ * from Frank Ch. Eigler <fche@xxxxxxxxxx>.
+ * – Thanks to Jeremy Fitzhardinge <jeremy@xxxxxxxx> for his constructive
+ * criticism about gcc optimization related issues.
+ *
+ * The marker mechanism supports multiple instances of the same marker.
+ * Markers can be put in inline functions, inlined static functions and
+ * unrolled loops.
+ *
+ * (C) Copyright 2006 Mathieu Desnoyers <mathieu.desnoyers@xxxxxxxxxx>
+ *
+ * This file is released under the GPLv2.
+ * See the file COPYING for more details.
+ */
+
+#ifndef __ASSEMBLY__
+
+typedef void marker_probe_func(const char *fmt, ...);
+
+#ifndef CONFIG_MARKERS_DISABLE_OPTIMIZATION
+#include <asm/marker.h>
+#else
+#include <asm-generic/marker.h>
+#endif
+
+#define MARK_NOARGS " "
+#define MARK_MAX_FORMAT_LEN 1024
+
+#ifndef CONFIG_MARKERS
+#define MARK(name, format, args...) \
+ __mark_check_format(format, ## args)
+#endif
+
+static inline __attribute__((format (printf, 1, 2)))
+void __mark_check_format(const char *fmt, ...)
+{ }
+
+extern marker_probe_func __mark_empty_function;
+
+extern int marker_set_probe(const char *name, const char *format,
+ marker_probe_func *probe);
+
+extern int marker_remove_probe(marker_probe_func *probe);
+extern int marker_list_probe(marker_probe_func *probe);
+
+#endif
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+#endif
diff --git a/include/linux/module.h b/include/linux/module.h
index eaec13d..5689857 100644
--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -254,6 +254,8 @@ struct module
const struct kernel_symbol *syms;
unsigned int num_syms;
const unsigned long *crcs;
+ const struct __mark_marker *markers;
+ unsigned int num_markers;

/* GPL-only exported symbols. */
const struct kernel_symbol *gpl_syms;
diff --git a/kernel/Kconfig.marker b/kernel/Kconfig.marker
new file mode 100644
index 0000000..deb237d
--- /dev/null
+++ b/kernel/Kconfig.marker
@@ -0,0 +1,17 @@
+# Code markers configuration
+
+config MARKERS
+ bool "Activate markers"
+ select MODULES
+ default n
+ help
+ Place an empty function call at each marker site. Can be
+ dynamically changed for a probe function.
+
+config MARKERS_DISABLE_OPTIMIZATION
+ bool "Disable architecture specific marker optimization"
+ depends EMBEDDED
+ default n
+ help
+ Disable code replacement jump optimisations. Especially useful if your
+ code is in a read-only rom/flash.
diff --git a/kernel/module.c b/kernel/module.c
index bbe0486..de80b76 100644
--- a/kernel/module.c
+++ b/kernel/module.c
@@ -123,6 +123,8 @@ extern const struct kernel_symbol __stop
extern const unsigned long __start__kcrctab[];
extern const unsigned long __start__kcrctab_gpl[];
extern const unsigned long __start__kcrctab_gpl_future[];
+extern const struct __mark_marker __start__markers[];
+extern const struct __mark_marker __stop__markers[];

#ifdef CONFIG_MODVERSIONS
#define symversion(base, idx) NULL
@@ -236,6 +238,170 @@ static struct module *find_module(const
```

```

return NULL;
}

+#ifdef CONFIG_MARKERS
+void __mark_empty_function(const char *fmt, ...)
+{
+}
+EXPORT_SYMBOL(__mark_empty_function);
+
+#define MARK_ENABLE_OFFSET(a) \
+(typeof(a))((char*)a+MARK_ENABLE_IMMEDIATE_OFFSET)
+
+static int marker_set_probe_range(const char *name,
+ const char *format,
+ marker_probe_func *probe,
+ const struct __mark_marker *begin,
+ const struct __mark_marker *end)
+{
+ const struct __mark_marker *iter;
+ int found = 0;
+
+ for(iter = begin; iter < end; iter++) {
+ if (strcmp(name, iter->cmark->name) == 0) {
+ if (format
+ && strcmp(format, iter->cmark->format) != 0) {
+ printk(KERN_NOTICE
+ "Format mismatch for probe %s "
+ "(%s), marker (%s)\n",
+ name,
+ format,
+ iter->cmark->format);
+ continue;
+ }
+ if (probe == __mark_empty_function) {
+ if (*iter->cmark->call
+ != __mark_empty_function) {
+ *iter->cmark->call =
+ __mark_empty_function;
+ }
+ /* Can have many enables for one function */
+ *MARK_ENABLE_OFFSET(iter->enable) = 0;
+ } else {
+ if (*iter->cmark->call
+ != __mark_empty_function) {
+ if (*iter->cmark->call != probe) {
+ printk(KERN_NOTICE
+ "Marker %s busy, "
+ "probe %p already "
+ "installed\n",
+ name,
+ *iter->cmark->call);

```

```

+ continue;
+ }
+ } else {
+ found++;
+ *iter->cmark->call = probe;
+ }
+ /* Can have many enables for one function */
+ *MARK_ENABLE_OFFSET(iter->enable) = 1;
+ }
+ found++;
+ }
+ }
+ return found;
+}
+
+static int marker_remove_probe_range(marker_probe_func *probe,
+ const struct __mark_marker *begin,
+ const struct __mark_marker *end)
+{
+ const struct __mark_marker *iter;
+ int found = 0;
+
+ for(iter = begin; iter < end; iter++) {
+ if (*iter->cmark->call == probe) {
+ *MARK_ENABLE_OFFSET(iter->enable) = 0;
+ *iter->cmark->call = __mark_empty_function;
+ found++;
+ }
+ }
+ return found;
+}
+
+static int marker_list_probe_range(marker_probe_func *probe,
+ const struct __mark_marker *begin,
+ const struct __mark_marker *end)
+{
+ const struct __mark_marker *iter;
+ int found = 0;
+
+ for(iter = begin; iter < end; iter++) {
+ if (probe)
+ if (probe != *iter->cmark->call) continue;
+ printk("name %s \n", iter->cmark->name);
+ printk(" enable %u ", *MARK_ENABLE_OFFSET(iter->enable));
+ printk(" func 0x%p format \"%s\"\n",
+ *iter->cmark->call, iter->cmark->format);
+ found++;
+ }
+ return found;
+}
+/* markers use the modlist_lock to to synchronise */

```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+int marker_set_probe(const char *name, const char *format,
+ marker_probe_func *probe)
+{
+ struct module *mod;
+ int found = 0;
+ unsigned long flags;
+
+ spin_lock_irqsave(&modlist_lock, flags);
+ /* Core kernel markers */
+ found += marker_set_probe_range(name, format, probe,
+ __start__markers, __stop__markers);
+ /* Markers in modules. */
+ list_for_each_entry(mod, &modules, list) {
+ found += marker_set_probe_range(name, format, probe,
+ mod->markers, mod->markers+mod->num_markers);
+ }
+ spin_unlock_irqrestore(&modlist_lock, flags);
+ return found;
+}
+EXPORT_SYMBOL(marker_set_probe);
+
+int marker_remove_probe(marker_probe_func *probe)
+{
+ struct module *mod;
+ int found = 0;
+ unsigned long flags;
+
+ spin_lock_irqsave(&modlist_lock, flags);
+ /* Core kernel markers */
+ found += marker_remove_probe_range(probe,
+ __start__markers, __stop__markers);
+ /* Markers in modules. */
+ list_for_each_entry(mod, &modules, list) {
+ found += marker_remove_probe_range(probe,
+ mod->markers, mod->markers+mod->num_markers);
+ }
+ spin_unlock_irqrestore(&modlist_lock, flags);
+ return found;
+}
+EXPORT_SYMBOL(marker_remove_probe);
+
+int marker_list_probe(marker_probe_func *probe)
+{
+ struct module *mod;
+ int found = 0;
+ unsigned long flags;
+
+ spin_lock_irqsave(&modlist_lock, flags);
+ /* Core kernel markers */
+ printk("Listing kernel markers\n");
+ found += marker_list_probe_range(probe,
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ __start__markers, __stop__markers);
+ /* Markers in modules. */
+ printk("Listing module markers\n");
+ list_for_each_entry(mod, &modules, list) {
+ printk("Listing markers for module %s\n", mod->name);
+ found += marker_list_probe_range(probe,
+ mod->markers, mod->markers+mod->num_markers);
+ }
+ spin_unlock_irqrestore(&modlist_lock, flags);
+ return found;
+}
+EXPORT_SYMBOL(marker_list_probe);
+#endif
+
+#ifdef CONFIG_SMP
+/* Number of blocks used and allocated. */
+static unsigned int pcpu_num_used, pcpu_num_allocated;
+@@ -1412,7 +1578,7 @@ static struct module *load_module(void _
+unsigned int i, symindex = 0, strindex = 0, setupindex, exindex,
+exportindex, modindex, obsparindex, infoindex, gplindex,
+crcindex, gplcrcindex, versindex, pcpuindex, gplfutureindex,
+- gplfuturecrcindex;
+ + gplfuturecrcindex, markersindex;
+struct module *mod;
+long err = 0;
+void *percpu = NULL, *ptr = NULL; /* Stops spurious gcc warning */
+@@ -1502,6 +1668,7 @@ #endif
+versindex = find_sec(hdr, sechdrs, secstrings, "__versions");
+infoindex = find_sec(hdr, sechdrs, secstrings, ".modinfo");
+pcpuindex = find_pcpusec(hdr, sechdrs, secstrings);
+ + markersindex = find_sec(hdr, sechdrs, secstrings, ".markers");
+
+/* Don't keep modinfo section */
+sechdrs[infoindex].sh_flags &= ~(unsigned long)SHF_ALLOC;
+@@ -1510,6 +1677,11 @@ #ifdef CONFIG_KALLSYMS
+sechdrs[symindex].sh_flags |= SHF_ALLOC;
+sechdrs[strindex].sh_flags |= SHF_ALLOC;
+#endif
+
+#ifdef CONFIG_MARKERS
+ + sechdrs[markersindex].sh_flags |= SHF_ALLOC;
+
+#else
+ + sechdrs[markersindex].sh_flags &= ~(unsigned long)SHF_ALLOC;
+
+#endif
+
+/* Check module struct version now, before we try to use module. */
+if (!check_modstruct_version(sechdrs, versindex, mod)) {
+@@ -1642,6 +1814,11 @@ #endif
+mod->gpl_future_syms = (void *)sechdrs[gplfutureindex].sh_addr;
+if (gplfuturecrcindex)
+mod->gpl_future_crcs = (void *)sechdrs[gplfuturecrcindex].sh_addr;
+ + if (markersindex) {
```

[PATCH] Linux Kernel Markers 0.20 for 2.6.17

```
+ mod->markers = (void *)sechdrs[markersindex].sh_addr;
+ mod->num_markers =
+ sechdrs[markersindex].sh_size / sizeof(*mod->markers);
+ }
```

```
#ifdef CONFIG_MODVERSIONS
if ((mod->num_syms && !crcindex) ||
```

```
---END---
```

OpenPGP public key: <http://krystal.dyndns.org:8080/key/compuj.gpg>

Key fingerprint: 8CD5 52C3 8E3C 4140 715F BA06 3F25 A8FE 3BAE 9A68

-

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/>