

Re: [PATCH] markers: modpost

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-11/msg00055.html>

- *From:* Mathieu Desnoyers <mathieu.desnoyers@xxxxxxxxxx>
 - *Date:* Wed, 31 Oct 2007 22:46:22 -0400
-

* Roland McGrath (roland@xxxxxxxxxx) wrote:

This adds some new magic in the MODPOST phase for CONFIG_MARKERS. Analogous to the Module.symvers file, the build will now write a Module.markers file when CONFIG_MARKERS=y is set. This file lists the name, defining module, and format string of each marker, separated by \t characters. This simple text file can be used by offline build procedures for instrumentation code, analogous to how System.map and Module.symvers can be useful to have for kernels other than the one you are running right now.

The method of extracting the strings is somewhat crude, but is very simple and should work fine in practice for the foreseeable future.

Hi Roland,

I'm ok with the idea of extracting such information, but I doubt one can assume the strings will always be layed out in the same order in the __markers_strings section. We also shouldn't assume that a marker name will be a neighbor of its format string.

If we want to do it safely, I think we should iterate from __start__markers to __stop__markers symbols of vmlinux and get the pointers to the name/format string pairs.

The same can then be done with modules using the __markers section.

Or maybe is there some reason not to do that ?

Mathieu

Signed-off-by: Roland McGrath <roland@xxxxxxxxxx>

scripts/Makefile.modpost | 11 +++

Re: [PATCH] markers: modpost

```
scripts/mod/modpost.c | 174 ++++++
scripts/mod/modpost.h | 3 +
3 files changed, 187 insertions(+), 1 deletions(-)
```

```
diff --git a/scripts/Makefile.modpost b/scripts/Makefile.modpost
index d988f5d..6321870 100644
--- a/scripts/Makefile.modpost
+++ b/scripts/Makefile.modpost
@@ -13,6 +13,7 @@
# 2) modpost is then used to
# 3) create one <module>.mod.c file pr. module
# 4) create one Module.symvers file with CRC for all exported symbols
+# 4a) [CONFIG_MARKERS] create one Module.markers file listing defined markers
# 5) compile all <module>.mod.c files
# 6) final link of the module to a <module.ko> file
```

```
@@ -45,6 +46,10 @@ include scripts/Makefile.lib
```

```
kernelsoymfile := $(objtree)/Module.symvers
modulesymfile := $(firstword $(KBUILD_EXTMOD))/Module.symvers
+kernelmarkersfile := $(objtree)/Module.markers
+modulemarkersfile := $(firstword $(KBUILD_EXTMOD))/Module.markers
+
+markersfile = $(if $(KBUILD_EXTMOD),$(modulemarkersfile),$(kernelmarkersfile))
```

```
# Step 1), find all modules listed in $(MODVERDIR)/
__modules := $(sort $(shell grep -h '\.ko' /dev/null $(wildcard $(MODVERDIR)/*.mod)))
@@ -62,6 +67,8 @@ modpost = scripts/mod/modpost \
$(if $(KBUILD_EXTMOD),-i,-o) $(kernelsoymfile) \
$(if $(KBUILD_EXTMOD),-I $(modulesymfile)) \
$(if $(KBUILD_EXTMOD),-o $(modulesymfile)) \
+ $(if $(CONFIG_MARKERS),-K $(kernelmarkersfile)) \
+ $(if $(CONFIG_MARKERS),-M $(markersfile)) \
$(if $(KBUILD_EXTMOD)$(KBUILD_MODPOST_WARN),-w)
```

```
quiet_cmd_modpost = MODPOST $(words $(filter-out vmlinux FORCE, $^)) modules
@@ -81,6 +88,10 @@ vmlinux.o: FORCE
$(symverfile): __modpost ;
$(modules:.ko=.mod.c): __modpost ;
```

```
+ifdef CONFIG_MARKERS
+$(markersfile): __modpost ;
+endif
+
```

```
# Step 5), compile all *.mod.c files
```

```
diff --git a/scripts/mod/modpost.c b/scripts/mod/modpost.c
index 93ac52a..df80bfc 100644
--- a/scripts/mod/modpost.c
+++ b/scripts/mod/modpost.c
```

Re: [PATCH] markers: modpost

```
@@ -11,6 +11,8 @@
* Usage: modpost vmlinux module1.o module2.o ...
*/

+#define _GNU_SOURCE
+#include <stdio.h>
#include <ctype.h>
#include "modpost.h"
#include "../include/linux/license.h"
@@ -424,6 +426,8 @@ static int parse_elf(struct elf_info *info, const char *filename)
info->export_unused_gpl_sec = i;
else if (strcmp(secname, "__ksymtab_gpl_future") == 0)
info->export_gpl_future_sec = i;
+ else if (strcmp(secname, "__markers_strings") == 0)
+ info->markers_strings_sec = i;

if (sechdrs[i].sh_type != SHT_SYMTAB)
continue;
@@ -1249,6 +1253,73 @@ static int exit_section_ref_ok(const char *name)
return 0;
}

+static size_t strlen_with_padding(const char *start, const char *limit)
+{
+ const char *p = memchr(start, '\0', limit - start);
+ if (p == NULL)
+ return 0;
+ do
+ ++p;
+ while (p < limit && *p == '\0');
+ return p - start;
+}
+
+static void get_markers(struct elf_info *info, struct module *mod)
+{
+ const Elf_Shdr *sh = &info->sechdrs[info->markers_strings_sec];
+ const char *strings;
+ const char *strings_end;
+ const char *p;
+ size_t n, i;
+
+ if (!info->markers_strings_sec)
+ return;
+
+ strings = (const char *) info->hdr + sh->sh_offset;
+ strings_end = strings + sh->sh_size;
+
+ /*
+ * First count the strings. They come in pairs of name, format.
+ */
+ for (n = 0, p = strings; p < strings_end; ++n) {
```

Re: [PATCH] markers: modpost

```
+ size_t len = strlen_with_padding(p, strings_end);
+ if (len == 0)
+ break;
+ p += len;
+ }
+ if (n % 2 != 0 || p != strings_end) {
+ warn("%s.ko has bad __markers_strings, ignoring it\n",
+ mod->name);
+ return;
+ }
+
+ if (n == 0)
+ return;
+
+ /*
+ * Now collect each pair into a formatted line for the output.
+ * Lines look like:
+ * marker_name vmlinux marker %s format %d
+ * The format string after the second \t can use whitespace.
+ */
+ mod->markers = NOFAIL(malloc(sizeof mod->markers[0] * n / 2));
+ mod->nmarkers = n / 2;
+
+ p = strings;
+ for (i = 0; i < n; i += 2) {
+ const char *name, *fmt;
+ name = p;
+ p += strlen_with_padding(p, strings_end);
+ fmt = p;
+ p += strlen_with_padding(p, strings_end);
+
+ mod->markers[i / 2] = NULL;
+ asprintf(&mod->markers[i / 2], "%s\t%s\t%s\n",
+ name, mod->name, fmt);
+ NOFAIL(mod->markers[i / 2]);
+ }
+ }
+
+ static void read_symbols(char *modname)
+ {
+ const char *symname;
+ @@ -1301,6 +1372,8 @@ static void read_symbols(char *modname)
+ get_src_version(modname, mod->srcversion,
+ sizeof(mod->srcversion)-1);
+
+ get_markers(&info, mod);
+
+ parse_elf_finish(&info);
+
+ /* Our trick to get versioning for struct_module - it's
+ @@ -1649,6 +1722,91 @@ static void write_dump(const char *fname)
```

Re: [PATCH] markers: modpost

```
write_if_changed(&buf, fname);
}

+static void add_marker(struct module *mod, const char *name, const char *fmt)
+{
+ char *line = NULL;
+ asprintf(&line, "%s\t%s\t%s\n", name, mod->name, fmt);
+ NOFAIL(line);
+
+
+ mod->markers = NOFAIL(realloc(mod->markers, ((mod->nmarkers + 1) *
+ sizeof mod->markers[0])));
+ mod->markers[mod->nmarkers++] = line;
+}
+
+static void read_markers(const char *fname)
+{
+ unsigned long size, pos = 0;
+ void *file = grab_file(fname, &size);
+ char *line;
+
+
+ if (!file)
+ /* No old markers, silently ignore */
+ return;
+
+
+ while ((line = get_next_line(&pos, file, size))) {
+ char *marker, *modname, *fmt;
+ struct module *mod;
+
+
+ marker = line;
+ if (!(modname = strchr(marker, '\t')))
+ goto fail;
+ *modname++ = '\0';
+ if (!(fmt = strchr(modname, '\t')))
+ goto fail;
+ *fmt++ = '\0';
+ if (*marker == '\0' || *modname == '\0')
+ goto fail;
+
+
+ if (!(mod = find_module(modname))) {
+ if (is_vmlinux(modname)) {
+ have_vmlinux = 1;
+ }
+ mod = new_module(NOFAIL(strdup(modname)));
+ mod->skip = 1;
+ }
+
+
+ add_marker(mod, marker, fmt);
+ }
+ return;
+fail:
+ fatal("parse error in markers list file\n");
```

Re: [PATCH] markers: modpost

```
+}
+
+static int compare_strings(const void *a, const void *b)
+{
+ return strcmp(*(const char **) a, *(const char **) b);
+}
+
+static void write_markers(const char *fname)
+{
+ struct buffer buf = { };
+ struct module *mod;
+ size_t i;
+
+ for (mod = modules; mod; mod = mod->next)
+ if ((!external_module || !mod->skip) && mod->markers != NULL) {
+ /*
+ * Sort the strings so we can skip duplicates when
+ * we write them out.
+ */
+ qsort(mod->markers, mod->nmarkers,
+ sizeof mod->markers[0], &compare_strings);
+ for (i = 0; i < mod->nmarkers; ++i) {
+ char *line = mod->markers[i];
+ buf_write(&buf, line, strlen(line));
+ while (i + 1 < mod->nmarkers &&
+ !strcmp(mod->markers[i],
+ mod->markers[i + 1]))
+ free(mod->markers[i++]);
+ free(mod->markers[i]);
+ }
+ free(mod->markers);
+ mod->markers = NULL;
+ }
+
+ write_if_changed(&buf, fname);
+}
+
+int main(int argc, char **argv)
+{
+ struct module *mod;
+ @@ -1656,10 +1814,12 @@ int main(int argc, char **argv)
+ char fname[SZ];
+ char *kernel_read = NULL, *module_read = NULL;
+ char *dump_write = NULL;
+ char *markers_read = NULL;
+ char *markers_write = NULL;
+ int opt;
+ int err;
+
+ while ((opt = getopt(argc, argv, "i:I:mso:aw")) != -1) {
+ while ((opt = getopt(argc, argv, "i:I:mso:awM:K:")) != -1) {
```

Re: [PATCH] markers: modpost

```
switch(opt) {
case 'i':
kernel_read = optarg;
@@ -1683,6 +1843,12 @@ int main(int argc, char **argv)
case 'w':
warn_unresolved = 1;
break;
+ case 'M':
+ markers_write = optarg;
+ break;
+ case 'K':
+ markers_read = optarg;
+ break;
default:
exit(1);
}
@@ -1724,5 +1890,11 @@ int main(int argc, char **argv)
if (dump_write)
write_dump(dump_write);

+ if (markers_read)
+ read_markers(markers_read);
+
+ if (markers_write)
+ write_markers(markers_write);
+
return err;
}
diff --git a/scripts/mod/modpost.h b/scripts/mod/modpost.h
index 0ffed17..175301a 100644
--- a/scripts/mod/modpost.h
+++ b/scripts/mod/modpost.h
@@ -110,6 +110,8 @@ struct module {
int has_init;
int has_cleanup;
struct buffer dev_table_buf;
+ char **markers;
+ size_t nmarkers;
char srcversion[25];
};

@@ -124,6 +126,7 @@ struct elf_info {
Elf_Section export_gpl_sec;
Elf_Section export_unused_gpl_sec;
Elf_Section export_gpl_future_sec;
+ Elf_Section markers_strings_sec;
const char *strtab;
char *modinfo;
unsigned int modinfo_len;
```

Re: [PATCH] markers: modpost

Mathieu Desnoyers

Computer Engineering Ph.D. Student, Ecole Polytechnique de Montreal

OpenPGP 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@xxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>