Subject: [PATCH] fail kernel compilation in case of unresolved symbols (v2)
Posted by Kirill Korotaev on Thu, 07 Sep 2006 09:59:46 GMT
View Forum Message <> Reply to Message

At stage 2 modpost utility is used to check modules.
In case of unresolved symbols modpost only prints warning.

IMHO it is a good idea to fail compilation process in case of
unresolved symbols (at least in modules coming with kernel),
since usually such errors are left unnoticed, but kernel
modules are broken.

Changes from v1:
- new option '-w' is added to modpost:
  if option is specified, modpost only warns about unresolved symbols
- modpost is called with '-w' for external modules in Makefile.modpost

Signed-Off-By: Andrey Mirkin <amirkin@sw.ru>
Signed-Off-By: Kirill Korotaev <dev@openvz.org>

```
diff --git a/scripts/Makefile.modpost b/scripts/Makefile.modpost
index 0a64688..9c01886 100644
--- a/scripts/Makefile.modpost
+++ b/scripts/Makefile.modpost
@@ -58,6 +58,7 @@ quiet_cmd_modpost = MODPOST
   $(if $(KBUILD_EXTMOD),-i,-o) $(kernelsymfile) \
   $(if $(KBUILD_EXTMOD),-I $(modulesymfile)) \
   $(if $(KBUILD_EXTMOD),-o $(modulesymfile)) \
+ $(if $(KBUILD_EXTMOD),-w) \
   $(filter-out FORCE,$^)

  PHONY += __modpost
diff --git a/scripts/mod/modpost.c b/scripts/mod/modpost.c
index dfde0e8..083a75e 100644
--- a/scripts/mod/modpost.c
+++ b/scripts/mod/modpost.c
@@ -23,6 +23,8 @@ int have_vmlinux = 0;
 static int all_versions = 0;
 /* If we are modposting external module set to 1 */
 static int external_module = 0;
+/* Only warn about unresolved symbols */
+static int warn_unresolved = 0;
 /* How a symbol is exported */
 enum export {
   export_plain,      export_unused,     export_gpl,
@@ -1187,16 +1189,19 @@ static void add_header(struct buffer *b,
```

```
 /**
  * Record CRCs for unresolved symbols
  **/
-static void add_versions(struct buffer *b, struct module *mod)
+static int add_versions(struct buffer *b, struct module *mod)
 {
  struct symbol *s, *exp;
+ int err = 0;

  for (s = mod->unres; s; s = s->next) {
   exp = find_symbol(s->name);
   if (!exp || exp->module == mod) {
-   if (have_vmlinux && !s->weak)
+   if (have_vmlinux && !s->weak) {
     warn("\"%s\" [%s.ko] undefined!\n",
          s->name, mod->name);
+    err = warn_unresolved ? 0 : 1;
+   }
    continue;
   }
   s->module = exp->module;
@@ -1205,7 +1210,7 @@ static void add_versions(struct buffer *
  }

  if (!modversions)
-  return;
+  return err;

  buf_printf(b, "\n");
  buf_printf(b, "static const struct modversion_info ____versions[]\n");
@@ -1225,6 +1230,8 @@ static void add_versions(struct buffer *
  }

  buf_printf(b, "};\n");
+
+ return err;
 }

 static void add_depends(struct buffer *b, struct module *mod,
@@ -1402,8 +1409,9 @@ int main(int argc, char **argv)
  char *kernel_read = NULL, *module_read = NULL;
  char *dump_write = NULL;
  int opt;
+ int err;

- while ((opt = getopt(argc, argv, "i:I:mo:a")) != -1) {
+ while ((opt = getopt(argc, argv, "i:I:mo:aw")) != -1) {
   switch(opt) {
```

```
    case 'i':
     kernel_read = optarg;
@@ -1421,6 +1429,9 @@ int main(int argc, char **argv)
    case 'a':
     all_versions = 1;
     break;
+   case 'w':
+    warn_unresolved = 1;
+    break;
    default:
     exit(1);
   }
@@ -1441,6 +1452,8 @@ int main(int argc, char **argv)
   check_exports(mod);
  }

+ err = 0;
+
  for (mod = modules; mod; mod = mod->next) {
   if (mod->skip)
    continue;
@@ -1448,7 +1461,7 @@ int main(int argc, char **argv)
   buf.pos = 0;

   add_header(&buf, mod);
-  add_versions(&buf, mod);
+  err |= add_versions(&buf, mod);
   add_depends(&buf, mod, modules);
   add_moddevtable(&buf, mod);
   add_srcversion(&buf, mod);
@@ -1460,5 +1473,5 @@ int main(int argc, char **argv)
  if (dump_write)
   write_dump(dump_write);

- return 0;
+ return err;
 }
```