Subject: [PATCH 1/2] Use list_head in binfmt handling
Posted by Alexey Dobriyan on Mon, 16 Jul 2007 13:26:33 GMT
View Forum Message <> Reply to Message

Switch single-linked binfmt formats list to usual list_head's.
This leads to one-liners in register_binfmt() and unregister_binfmt().
The downside is one pointer more in struct linux_binfmt. This is not a
problem, since the set of registered binfmts on typical box is very small
(ELF + something distro enabled for you).

Test-booted, played with executable .txt files, modprobe/rmmod binfmt_misc.

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>
---

 fs/exec.c | 34 ++++++----------------------------
 include/linux/binfmts.h | 3 +--
 2 files changed, 7 insertions(+), 30 deletions(-)

```
--- a/fs/exec.c
+++ b/fs/exec.c
@@ -66,27 +66,15 @@ int suid_dumpable = 0;
 EXPORT_SYMBOL(suid_dumpable);
 /* The maximal length of core_pattern is also specified in sysctl.c */

-static struct linux_binfmt *formats;
+static LIST_HEAD(formats);
 static DEFINE_RWLOCK(binfmt_lock);

 int register_binfmt(struct linux_binfmt * fmt)
 {
- struct linux_binfmt ** tmp = &formats;
-
  if (!fmt)
   return -EINVAL;
- if (fmt->next)
-  return -EBUSY;
  write_lock(&binfmt_lock);
- while (*tmp) {
-  if (fmt == *tmp) {
-   write_unlock(&binfmt_lock);
-   return -EBUSY;
-  }
-  tmp = &(*tmp)->next;
- }
- fmt->next = formats;
- formats = fmt;
+ list_add(&fmt->lh, &formats);
```

```
  write_unlock(&binfmt_lock);
  return 0;
 }
@@ -95,20 +83,10 @@ EXPORT_SYMBOL(register_binfmt);

 int unregister_binfmt(struct linux_binfmt * fmt)
 {
- struct linux_binfmt ** tmp = &formats;
-
  write_lock(&binfmt_lock);
- while (*tmp) {
-  if (fmt == *tmp) {
-   *tmp = fmt->next;
-   fmt->next = NULL;
-   write_unlock(&binfmt_lock);
-   return 0;
-  }
-  tmp = &(*tmp)->next;
- }
+ list_del(&fmt->lh);
  write_unlock(&binfmt_lock);
- return -EINVAL;
+ return 0;
 }

 EXPORT_SYMBOL(unregister_binfmt);
@@ -155,7 +133,7 @@ asmlinkage long sys_uselib(const char __user * library)
   struct linux_binfmt * fmt;

   read_lock(&binfmt_lock);
-  for (fmt = formats ; fmt ; fmt = fmt->next) {
+  list_for_each_entry(fmt, &formats, lh) {
    if (!fmt->load_shlib)
     continue;
    if (!try_module_get(fmt->module))
@@ -1097,7 +1075,7 @@ int search_binary_handler(struct linux_binprm *bprm,struct pt_regs
*regs)
  retval = -ENOENT;
  for (try=0; try<2; try++) {
   read_lock(&binfmt_lock);
-  for (fmt = formats ; fmt ; fmt = fmt->next) {
+  list_for_each_entry(fmt, &formats, lh) {
    int (*fn)(struct linux_binprm *, struct pt_regs *) = fmt->load_binary;
    if (!fn)
     continue;
--- a/include/linux/binfmts.h
+++ b/include/linux/binfmts.h
@@ -55,7 +55,7 @@ struct linux_binprm{
```

```
  * linux accepts.
  */
 struct linux_binfmt {
- struct linux_binfmt * next;
+ struct list_head lh;
  struct module *module;
  int (*load_binary)(struct linux_binprm *, struct  pt_regs * regs);
  int (*load_shlib)(struct file *);
```
