

[patch] Re: dlopen() and dlclose() are not MT-safe?

[patch] Re: dlopen() and dlclose() are not MT-safe?

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/hackers/2006-03/msg00226.html>

- *From:* Kostik Belousov <kostikbel@xxxxxxxxxx>
 - *Date:* Thu, 23 Mar 2006 12:54:40 +0200
-

On Thu, Mar 23, 2006 at 05:57:24AM +0900, Kazuaki Oda wrote:

Kostik Belousov wrote:

Oops. Completely reversed condition in the if. :(Also, I don't think it shall returns the error in this situation. New take:

Index: libexec/rtld-elf/rtld.c

```
=====
RCS file: /usr/local/arch/ncvs/src/libexec/rtld-elf/rtld.c,v
retrieving revision 1.112
diff -u -r1.112 rtld.c
--- libexec/rtld-elf/rtld.c 24 Dec 2005 15:37:30 -0000 1.112
+++ libexec/rtld-elf/rtld.c 22 Mar 2006 19:03:12 -0000
@@ -1688,6 +1688,11 @@
wlock_release(rtld_bind_lock, lockstate);
objlist_call_fini(&list_fini);
lockstate = wlock_acquire(rtld_bind_lock);
+ if (root->refcount != 0) {
+ wlock_release(rtld_bind_lock, lockstate);
+ return 0;
+ }
+
objlist_remove_unref(&list_fini);

/* Finish cleaning up the newly-unreferenced objects. */
```

Thanks. I applied your patch and ran the test program once more.

When linked with libpthread:
% ./dltest
Segmentation fault (core dumped)

When linked with libthr:
% ./dltest
ld-elf.so.1: assert failed: /usr/src/libexec/rtld-elf/rtld.c:1728

[patch] Re: dlopen() and dlclose() are not MT-safe?

In libpthread case, the result was changed. I got no "assert failed" message, and it took longer time to crash than before. In libthr case, I could not find any difference.

BTW do you know the reason why lock is released before calling objlist_call_fini()? If we don't release the lock, what problem will occur? deadlock?

The reasoning behind releasing the lock is to allow calls to dl*() functions from constructors/destructors. This is common practice and shall be supported. Yes, leaving the lock taken will lead to deadlock.

Please, try the following patch and report results. I can run (modified *) version of your test for some time without crash with both libpthread and libthr.

[* You need to allow for errors in dlopen and just continue the loop.]

Patch protects access to the list of unloading objects by bind lock, and marks the objects that are running finalizers to prevent simultaneous loading of them.

Also, it comments out code that is completely incomprehensible by my sloppy brain. Namely, the thread_mask_set stuff seems to allow for the thread to run as-is if another thread taken the lock. As result, lock is effectively ignored. I cannot understand purpose of this fragments. Hope, kan@ describe the reasons.

Index: libexec/rtld-elf/rtld.c

```
=====
RCS file: /usr/local/arch/ncvs/src/libexec/rtld-elf/rtld.c,v
retrieving revision 1.112
diff -u -r1.112 rtld.c
--- libexec/rtld-elf/rtld.c 24 Dec 2005 15:37:30 -0000 1.112
+++ libexec/rtld-elf/rtld.c 23 Mar 2006 10:44:27 -0000
@@ -105,7 +105,7 @@
static int load_preload_objects(void);
static Obj_Entry *load_object(const char *, const Obj_Entry *);
static Obj_Entry *obj_from_addr(const void *);
-static void objlist_call_fini(Objlist *);
+static void objlist_call_fini(Objlist *, int *lockstate, unsigned long *gen);
static void objlist_call_init(Objlist *);
static void objlist_clear(Objlist *);
static Objlist_Entry *objlist_find(Objlist *, const Obj_Entry *);
@@ -165,6 +165,7 @@
STAILQ_HEAD_INITIALIZER(list_main);
static Objlist list_fini = /* Objects needing fini() calls */
STAILQ_HEAD_INITIALIZER(list_fini);
+static unsigned long list_fini_gen = 0;
```

[patch] Re: dlopen() and dlclose() are not MT-safe?

```
static Elf_Sym sym_zero; /* For resolving undefined weak refs. */

@@ -1168,8 +1169,10 @@
objlist_push_tail(list, obj);

/* Add the object to the global fini list in the reverse order. */
- if (obj->fini != (Elf_Addr)NULL)
+ if (obj->fini != (Elf_Addr)NULL) {
objlist_push_head(&list_fini, obj);
+ list_fini_gen++;
+ }
}

#ifdef FPTR_TARGET
@@ -1362,21 +1365,30 @@
* non-NULL fini functions.
*/
static void
objlist_call_fini(Objlist *list)
+objlist_call_fini(Objlist *list, int *lockstate, unsigned long *gen)
{
- Objlist_Entry *elm;
+ Objlist_Entry *elm, *elm_tmp;
char *saved_msg;
+ unsigned long saved_gen = *gen;

/*
* Preserve the current error message since a fini function might
* call into the dynamic linker and overwrite it.
*/
saved_msg = errmsg_save();
- STAILQ_FOREACH(elm, list, link) {
+ again:
+ STAILQ_FOREACH_SAFE(elm, list, link, elm_tmp) {
if (elm->obj->refcount == 0) {
dbg("calling fini function for %s at %p", elm->obj->path,
(void *)elm->obj->fini);
+ elm->obj->fini_now = true;
+ wlock_release(rtld_bind_lock, *lockstate);
call_initfini_pointer(elm->obj, elm->obj->fini);
+ *lockstate = wlock_acquire(rtld_bind_lock);
+ if (*gen != saved_gen) {
+ saved_gen = *gen;
+ goto again;
+ }
}
}
errmsg_restore(saved_msg);
@@ -1390,7 +1402,7 @@
static void
objlist_call_init(Objlist *list)
```

[patch] Re: dlopen() and dlclose() are not MT-safe?

```
{
- Objlist_Entry *elm;
+ Objlist_Entry *elm, *elm_tmp;
char *saved_msg;

/*
@@ -1398,7 +1410,7 @@
* call into the dynamic linker and overwrite it.
*/
saved_msg = errmsg_save();
- STAILQ_FOREACH(elm, list, link) {
+ STAILQ_FOREACH_SAFE(elm, list, link, elm_tmp) {
dbg("calling init function for %s at %p", elm->obj->path,
(void *)elm->obj->init);
call_initfini_pointer(elm->obj, elm->obj->init);
@@ -1563,15 +1575,18 @@
rtld_exit(void)
{
Obj_Entry *obj;
+ int lockstate;

dbg("rtld_exit()");
/* Clear all the reference counts so the fini functions will be called. */
+ lockstate = wlock_acquire(rtld_bind_lock);
for (obj = obj_list; obj != NULL; obj = obj->next)
obj->refcount = 0;
- objlist_call_fini(&list_fini);
+ objlist_call_fini(&list_fini, &lockstate, &list_fini_gen);
/* No need to remove the items from the list, since we are exiting. */
if (!libmap_disable)
lm_fini();
+ wlock_release(rtld_bind_lock, lockstate);
}

static void *
@@ -1685,10 +1700,10 @@
* The object is no longer referenced, so we must unload it.
* First, call the fini functions with no locks held.
*/
- wlock_release(rtld_bind_lock, lockstate);
- objlist_call_fini(&list_fini);
- lockstate = wlock_acquire(rtld_bind_lock);
+ objlist_call_fini(&list_fini, &lockstate, &list_fini_gen);
+
objlist_remove_unref(&list_fini);
+ list_fini_gen++;

/* Finish cleaning up the newly-unreferenced objects. */
GDB_STATE(RT_DELETE, &root->linkmap);
@@ -1741,9 +1756,9 @@
if (ld_tracing != NULL)
```

[patch] Re: dlopen() and dlclose() are not MT-safe?

[patch] Re: dlopen() and dlclose() are not MT-safe?

```
environ = (char **) *get_program_var_addr("environ");
```

```
+ lockstate = wlock_acquire(rtld_bind_lock);  
objlist_init(&initlist);
```

```
- lockstate = wlock_acquire(rtld_bind_lock);  
GDB_STATE(RT_ADD, NULL);
```

```
old_obj_tail = obj_tail;  
@@ -1755,7 +1770,10 @@  
obj = load_object(name, obj_main);  
}
```

```
- if (obj) {  
+ if (obj && obj->fini_now) {  
+ obj = NULL;  
+ _rtld_error("%s is running finalizers now", name);  
+ } else if (obj) {  
obj->dl_refcount++;  
if (mode & RTLD_GLOBAL && objlist_find(&list_global, obj) == NULL)  
objlist_push_tail(&list_global, obj);  
Index: libexec/rtld-elf/rtld.h
```

```
RCS file: /usr/local/arch/ncvs/src/libexec/rtld-elf/rtld.h,v
```

```
retrieving revision 1.37
```

```
diff -u -r1.37 rtld.h
```

```
--- libexec/rtld-elf/rtld.h 18 Dec 2005 19:43:32 -0000 1.37
```

```
+++ libexec/rtld-elf/rtld.h 23 Mar 2006 10:44:27 -0000
```

```
@@ -210,6 +210,7 @@
```

```
bool jmpslots_done; /* Already have relocated the jump slots */
```

```
bool init_done; /* Already have added object to init list */
```

```
bool tls_done; /* Already allocated offset for static TLS */
```

```
+ bool fini_now; /* Finalizer for dso currently runs */
```

```
struct link_map linkmap; /* for GDB and dlinfo() */
```

```
Objlist dldags; /* Object belongs to these dlopened DAGs (%) */
```

```
Index: libexec/rtld-elf/rtld_lock.c
```

```
RCS file: /usr/local/arch/ncvs/src/libexec/rtld-elf/rtld_lock.c,v
```

```
retrieving revision 1.3
```

```
diff -u -r1.3 rtld_lock.c
```

```
--- libexec/rtld-elf/rtld_lock.c 16 Nov 2004 20:45:51 -0000 1.3
```

```
+++ libexec/rtld-elf/rtld_lock.c 23 Mar 2006 10:44:27 -0000
```

```
@@ -183,22 +183,22 @@
```

```
int
```

```
rlock_acquire(rtld_lock_t lock)
```

```
{
```

```
- if (thread_mask_set(lock->mask)) {
```

```
+/* if (thread_mask_set(lock->mask)) {
```

```
dbg("rlock_acquire: recursed");
```

```
return (0);
```

[patch] Re: dlopen() and dlclose() are not MT-safe?

[patch] Re: dlopen() and dlclose() are not MT-safe?

```
}
- lockinfo.rlock_acquire(lock->handle);
+*/ lockinfo.rlock_acquire(lock->handle);
return (1);
}

int
wlock_acquire(rtld_lock_t lock)
{
- if (thread_mask_set(lock->mask)) {
+*/ if (thread_mask_set(lock->mask)) {
dbg("wlock_acquire: recursed");
return (0);
}
- lockinfo.wlock_acquire(lock->handle);
+*/ lockinfo.wlock_acquire(lock->handle);
return (1);
}
```

freebsd-hackers@xxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-hackers>

To unsubscribe, send any mail to "freebsd-hackers-unsubscribe@xxxxxxxxxxx"