

[PATCH] Introduce a handy list_first_entry macro

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-04/msg05893.html>

- *From:* Pavel Emelianov <xemul@xxxxx>
 - *Date:* Tue, 17 Apr 2007 15:18:56 +0400
-

There are many places in the kernel where the construction like

```
foo = list_entry(head->next, struct foo_struct, list);
```

are used.

The code might look more descriptive and neat if using the macro

```
list_first_entry(head, type, member) \  
list_entry((head)->next, type, member)
```

Here is the macro itself and the examples of its usage in the generic code. If it will turn out to be useful, I can prepare the set of patches to inject in into arch-specific code, drivers, networking, etc.

Signed-off-by: Pavel Emelianov <xemul@xxxxxxxxxxx>

Signed-off-by: Kirill Korotaev <dev@xxxxxxxxxxx>

```
diff --git a/fs/dquot.c b/fs/dquot.c  
index ac40503..e5cce5a 100644  
--- a/fs/dquot.c  
+++ b/fs/dquot.c  
@@ -474,7 +474,7 @@ int vfs_quota_sync(struct super_block *s  
spin_lock(&dq_list_lock);  
dirty = &dqopt->info[cnt].dq_dirty_list;  
while (!list_empty(dirty)) {  
- dquot = list_entry(dirty->next, struct dquot, dq_dirty);  
+ dquot = list_first_entry(dirty, struct dquot, dq_dirty);  
/* Dirty and inactive can be only bad dquot... */  
if (!test_bit(DQ_ACTIVE_B, &dquot->dq_flags)) {  
clear_dquot_dirty(dquot);  
diff --git a/fs/eventpoll.c b/fs/eventpoll.c  
index ae5dc99..1aad34e 100644  
--- a/fs/eventpoll.c  
+++ b/fs/eventpoll.c  
@@ -371,7 +371,7 @@ static void ep_unregister_pollwait(struc
```

[PATCH] Introduce a handy list_first_entry macro

```
if (nwait) {
while (!list_empty(lsthead)) {
- pwq = list_entry(lsthead->next, struct eppoll_entry, llink);
+ pwq = list_first_entry(lsthead, struct eppoll_entry, llink);

list_del_init(&pwq->llink);
remove_wait_queue(pwq->whead, &pwq->wait);
@@ -602,7 +602,7 @@ void eventpoll_release_file(struct file
mutex_lock(&epmutex);

while (!list_empty(lsthead)) {
- epi = list_entry(lsthead->next, struct epitem, flink);
+ epi = list_first_entry(lsthead, struct epitem, flink);

ep = epi->ep;
list_del_init(&epi->flink);
@@ -943,7 +943,7 @@ static int ep_send_events(struct eventpo
* read.
*/
for (eventcnt = 0; !list_empty(txlist) && eventcnt < maxevents;) {
- epi = list_entry(txlist->next, struct epitem, rdllink);
+ epi = list_first_entry(txlist, struct epitem, rdllink);
prefetch(epi->rdllink.next);

/*
diff --git a/fs/inode.c b/fs/inode.c
index 224fe4d..359f406 100644
--- a/fs/inode.c
+++ b/fs/inode.c
@@ -286,7 +286,7 @@ static void dispose_list(struct list_he
while (!list_empty(head)) {
struct inode *inode;

- inode = list_entry(head->next, struct inode, i_list);
+ inode = list_first_entry(head, struct inode, i_list);
list_del(&inode->i_list);

if (inode->i_data.nrpages)
diff --git a/fs/inotify.c b/fs/inotify.c
index f5099d8..7457501 100644
--- a/fs/inotify.c
+++ b/fs/inotify.c
@@ -509,7 +509,7 @@ void inotify_destroy(struct inotify_hand
mutex_unlock(&ih->mutex);
break;
}
- watch = list_entry(watches->next, struct inotify_watch, h_list);
+ watch = list_first_entry(watches, struct inotify_watch, h_list);
get_inotify_watch(watch);
mutex_unlock(&ih->mutex);
```

[PATCH] Introduce a handy list_first_entry macro

```
diff --git a/fs/namespace.c b/fs/namespace.c
index 213f7ab..4627e1d 100644
--- a/fs/namespace.c
+++ b/fs/namespace.c
@@ -518,7 +518,7 @@ void release_mounts(struct list_head *he
 {
 struct vfsmount *mnt;
 while (!list_empty(head)) {
- mnt = list_entry(head->next, struct vfsmount, mnt_hash);
+ mnt = list_first_entry(head, struct vfsmount, mnt_hash);
 list_del_init(&mnt->mnt_hash);
 if (mnt->mnt_parent != mnt) {
 struct dentry *dentry;
@@ -1196,7 +1196,7 @@ static void expire_mount_list(struct lis

while (!list_empty(graveyard)) {
LIST_HEAD(umounts);
- mnt = list_entry(graveyard->next, struct vfsmount, mnt_expire);
+ mnt = list_first_entry(graveyard, struct vfsmount, mnt_expire);
list_del_init(&mnt->mnt_expire);

/* don't do anything if the namespace is dead - all the
diff --git a/fs/pnode.c b/fs/pnode.c
index 56aacea..89940f2 100644
--- a/fs/pnode.c
+++ b/fs/pnode.c
@@ -59,7 +59,7 @@ static int do_make_slave(struct vfsmount
} else {
struct list_head *p = &mnt->mnt_slave_list;
while (!list_empty(p)) {
- slave_mnt = list_entry(p->next,
+ slave_mnt = list_first_entry(p,
struct vfsmount, mnt_slave);
list_del_init(&slave_mnt->mnt_slave);
slave_mnt->mnt_master = NULL;
diff --git a/include/linux/list.h b/include/linux/list.h
index fd59659..71318fc 100644
--- a/include/linux/list.h
+++ b/include/linux/list.h
@@ -425,6 +425,9 @@ static inline void list_splice_init_rcu(
#define list_entry(ptr, type, member) \
container_of(ptr, type, member)

+#define list_first_entry(ptr, type, member) \
+ list_entry((ptr)->next, type, member)
+
+
/**
 * list_for_each - iterate over a list
 * @pos: the &struct list_head to use as a loop cursor.
diff --git a/kernel/posix-cpu-timers.c b/kernel/posix-cpu-timers.c
index 657f776..1de710e 100644
```

[PATCH] Introduce a handy list_first_entry macro

[PATCH] Introduce a handy list_first_entry macro

```
--- a/kernel/posix-cpu-timers.c
+++ b/kernel/posix-cpu-timers.c
@@ -971,7 +971,7 @@ static void check_thread_timers(struct t
maxfire = 20;
tsk->it_prof_expires = cputime_zero;
while (!list_empty(timers)) {
- struct cpu_timer_list *t = list_entry(timers->next,
+ struct cpu_timer_list *t = list_first_entry(timers,
struct cpu_timer_list,
entry);
if (!--maxfire || cputime_lt(prof_ticks(tsk), t->expires.cpu)) {
@@ -986,7 +986,7 @@ static void check_thread_timers(struct t
maxfire = 20;
tsk->it_virt_expires = cputime_zero;
while (!list_empty(timers)) {
- struct cpu_timer_list *t = list_entry(timers->next,
+ struct cpu_timer_list *t = list_first_entry(timers,
struct cpu_timer_list,
entry);
if (!--maxfire || cputime_lt(virt_ticks(tsk), t->expires.cpu)) {
@@ -1001,7 +1001,7 @@ static void check_thread_timers(struct t
maxfire = 20;
tsk->it_sched_expires = 0;
while (!list_empty(timers)) {
- struct cpu_timer_list *t = list_entry(timers->next,
+ struct cpu_timer_list *t = list_first_entry(timers,
struct cpu_timer_list,
entry);
if (!--maxfire || tsk->sched_time < t->expires.sched) {
@@ -1057,7 +1057,7 @@ static void check_process_timers(struct
maxfire = 20;
prof_expires = cputime_zero;
while (!list_empty(timers)) {
- struct cpu_timer_list *t = list_entry(timers->next,
+ struct cpu_timer_list *t = list_first_entry(timers,
struct cpu_timer_list,
entry);
if (!--maxfire || cputime_lt(ptime, t->expires.cpu)) {
@@ -1072,7 +1072,7 @@ static void check_process_timers(struct
maxfire = 20;
virt_expires = cputime_zero;
while (!list_empty(timers)) {
- struct cpu_timer_list *t = list_entry(timers->next,
+ struct cpu_timer_list *t = list_first_entry(timers,
struct cpu_timer_list,
entry);
if (!--maxfire || cputime_lt(utime, t->expires.cpu)) {
@@ -1087,7 +1087,7 @@ static void check_process_timers(struct
maxfire = 20;
sched_expires = 0;
while (!list_empty(timers)) {
```

[PATCH] Introduce a handy list_first_entry macro

[PATCH] Introduce a handy list_first_entry macro

```
- struct cpu_timer_list *t = list_entry(timers->next,
+ struct cpu_timer_list *t = list_first_entry(timers,
struct cpu_timer_list,
entry);
if (!--maxfire || sched_time < t->expires.sched) {
@@ -1400,7 +1400,7 @@ void set_process_cpu_timer(struct task_s
*/
head = &tsk->signal->cpu_timers[clock_idx];
if (list_empty(head) ||
- cputime_ge(list_entry(head->next,
+ cputime_ge(list_first_entry(head,
struct cpu_timer_list, entry)->expires.cpu,
*newval)) {
/*
diff --git a/kernel/timer.c b/kernel/timer.c
index 2baf189..b990e95 100644
--- a/kernel/timer.c
+++ b/kernel/timer.c
@@ -627,7 +627,7 @@ static inline void __run_timers(tvec_base
void (*fn)(unsigned long);
unsigned long data;

- timer = list_entry(head->next, struct timer_list, entry);
+ timer = list_first_entry(head, struct timer_list, entry);
fn = timer->function;
data = timer->data;

@@ -1246,7 +1246,7 @@ static void migrate_timer_list(tvec_base
struct timer_list *timer;

while (!list_empty(head)) {
- timer = list_entry(head->next, struct timer_list, entry);
+ timer = list_first_entry(head, struct timer_list, entry);
detach_timer(timer, 0);
timer_set_base(timer, new_base);
internal_add_timer(new_base, timer);
-
```

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/>