Subject: [PATCH] Uninline fork.c/exit.c
Posted by Alexey Dobriyan on Mon, 01 Oct 2007 12:15:53 GMT
View Forum Message <> Reply to Message

Save ~650 bytes here.

add/remove: 4/0 grow/shrink: 0/7 up/down: 430/-1088 (-658)

function	old	new	delta
__copy_fs_struct	-	202	+202
__put_fs_struct	-	112	+112
__exit_fs	-	58	+58
__exit_files	-	58	+58
exit_files	58	2	-56
put_fs_struct	112	5	-107
exit_fs	161	2	-159
sys_unshare	774	590	-184
copy_process	4031	3840	-191
do_exit	1791	1597	-194
copy_fs_struct	202	5	-197

No difference in lmbench lat_proc tests on 2-way Opteron 246.
Smaaaal degradation on UP P4 (within errors).

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

 kernel/exit.c | 6 +++---
 kernel/fork.c | 20 ++++++++++++----------
 2 files changed, 13 insertions(+), 13 deletions(-)

```
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -493,7 +493,7 @@ void reset_files_struct(struct task_struct *tsk, struct files_struct *files)
 }
 EXPORT_SYMBOL(reset_files_struct);

-static inline void __exit_files(struct task_struct *tsk)
+static void __exit_files(struct task_struct *tsk)
 {
  struct files_struct * files = tsk->files;

@@ -510,7 +510,7 @@ void exit_files(struct task_struct *tsk)
  __exit_files(tsk);
 }

-static inline void __put_fs_struct(struct fs_struct *fs)
+static void __put_fs_struct(struct fs_struct *fs)
 {
```

```
  /* No need to hold fs->lock if we are killing it */
  if (atomic_dec_and_test(&fs->count)) {
@@ -531,7 +531,7 @@ void put_fs_struct(struct fs_struct *fs)
  __put_fs_struct(fs);
 }

-static inline void __exit_fs(struct task_struct *tsk)
+static void __exit_fs(struct task_struct *tsk)
 {
  struct fs_struct * fs = tsk->fs;

--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -195,7 +195,7 @@ static struct task_struct *dup_task_struct(struct task_struct *orig)
 }

 #ifdef CONFIG_MMU
-static inline int dup_mmap(struct mm_struct *mm, struct mm_struct *oldmm)
+static int dup_mmap(struct mm_struct *mm, struct mm_struct *oldmm)
 {
  struct vm_area_struct *mpnt, *tmp, **pprev;
  struct rb_node **rb_link, *rb_parent;
@@ -573,7 +573,7 @@ fail_nomem:
  return retval;
 }

-static inline struct fs_struct *__copy_fs_struct(struct fs_struct *old)
+static struct fs_struct *__copy_fs_struct(struct fs_struct *old)
 {
  struct fs_struct *fs = kmem_cache_alloc(fs_cachep, GFP_KERNEL);
  /* We don't need to lock fs - think why ;-) */
@@ -605,7 +605,7 @@ struct fs_struct *copy_fs_struct(struct fs_struct *old)

 EXPORT_SYMBOL_GPL(copy_fs_struct);

-static inline int copy_fs(unsigned long clone_flags, struct task_struct * tsk)
+static int copy_fs(unsigned long clone_flags, struct task_struct * tsk)
 {
  if (clone_flags & CLONE_FS) {
   atomic_inc(&current->fs->count);
@@ -808,7 +808,7 @@ int unshare_files(void)

 EXPORT_SYMBOL(unshare_files);

-static inline int copy_sighand(unsigned long clone_flags, struct task_struct * tsk)
+static int copy_sighand(unsigned long clone_flags, struct task_struct * tsk)
 {
  struct sighand_struct *sig;
```

```
@@ -831,7 +831,7 @@ void __cleanup_sighand(struct sighand_struct *sighand)
   kmem_cache_free(sighand_cachep, sighand);
 }

-static inline int copy_signal(unsigned long clone_flags, struct task_struct * tsk)
+static int copy_signal(unsigned long clone_flags, struct task_struct * tsk)
 {
  struct signal_struct *sig;
  int ret;
@@ -911,7 +911,7 @@ void __cleanup_signal(struct signal_struct *sig)
  kmem_cache_free(signal_cachep, sig);
 }

-static inline void cleanup_signal(struct task_struct *tsk)
+static void cleanup_signal(struct task_struct *tsk)
 {
  struct signal_struct *sig = tsk->signal;

@@ -921,7 +921,7 @@ static inline void cleanup_signal(struct task_struct *tsk)
   __cleanup_signal(sig);
 }

-static inline void copy_flags(unsigned long clone_flags, struct task_struct *p)
+static void copy_flags(unsigned long clone_flags, struct task_struct *p)
 {
  unsigned long new_flags = p->flags;

@@ -939,7 +939,7 @@ asmlinkage long sys_set_tid_address(int __user *tidptr)
  return current->pid;
 }

-static inline void rt_mutex_init_task(struct task_struct *p)
+static void rt_mutex_init_task(struct task_struct *p)
 {
  spin_lock_init(&p->pi_lock);
 #ifdef CONFIG_RT_MUTEXES
@@ -1338,7 +1338,7 @@ struct task_struct * __cpuinit fork_idle(int cpu)
  return task;
 }

-static inline int fork_traceflag (unsigned clone_flags)
+static int fork_traceflag (unsigned clone_flags)
 {
  if (clone_flags & CLONE_UNTRACED)
   return 0;
@@ -1468,7 +1468,7 @@ void __init proc_caches_init(void)
  * Check constraints on flags passed to the unshare system call and
```

```
  * force unsharing of additional process context as appropriate.
  */
-static inline void check_unshare_flags(unsigned long *flags_ptr)
+static void check_unshare_flags(unsigned long *flags_ptr)
 {
  /*
   * If unsharing a thread from a thread group, must also
```