
Subject: Re: [RFC][PATCH] rename 'struct pid'
Posted by [serue](#) on Wed, 11 Apr 2007 00:57:33 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quoting Dave Hansen (hansendc@us.ibm.com):

>
> I've been hacking quite a bit on the pidspace code. I've run
> into a bug or two, and had a heck of a time debugging it.
> Other than my inferior puny monkey brain, I'm directing some
> of the blame squarely in the direction of the 'struct pid'.
>
> We have pid_t, pid_ns, struct pid and pid_link, at _least_.
> Seeing code like get_pid(pid->pid_ns->pid_type[PIDTYPE_PID])
> is mind-numbing to say the least. It makes it really hard
> to comprehend, and even harder to debug.
>
> We honestly have a lot of code like this:
>
> pid = pid_nr(filp->f_owner.pid);
>
> WTF? It's getting a pid from a pid? Huh? :)
>
> It makes sense when you go look at the structures, but
> sitting in the middle of a function and when you can't see
> all of the struct declarations can be really sketchy.
>
> So, I propose that we rename the one structure that seems to
> be the focus of the problem: 'struct pid'. Fundamentally, it
> is a 'process identifier': it helps the kernel to identify
> processes. However, as I noted, 'pid' is a wee bit overloaded.
>
> In addition to "identifying" processes, this structure acts
> as an indirection or handle to them. So, I propose we call
> these things 'struct task_ref'. Just reading some of the

I haven't tested the patch, but I enthusiastically, wholeheartedly,
excitedly agree with this name change.

-serge

(I plan to run some cross-arch tests in the morning)

> code that I've modified in here makes me feel like this is
> the right way.
>
> Compare the two sentences below:
>
> Oh, I have a task_ref? What kind is it? Oh, it's a pgid

> reference because I have REFTYPE_PGID.
>
> Oh, I have a pid? What kind is it? Oh, it's a pid because
> I have PIDTYPE_PID.
>
> Which makes more sense?
>
> So, this still needs some work converting some of the
> function names, but it compiles as-is. Any ideas for better
> names?
>
> Signed-off-by: Dave Hansen <hansendc@us.ibm.com>
> ---
>
> lxc-dave/drivers/char/keyboard.c | 10 +-
> lxc-dave/drivers/char/n_r3964.c | 32 +++---
> lxc-dave/drivers/char/tty_io.c | 66 ++++++-----
> lxc-dave/drivers/char/vt.c | 2
> lxc-dave/drivers/char/vt_ioctl.c | 16 +--
> lxc-dave/drivers/net/tun.c | 2
> lxc-dave/drivers/s390/char/fs3270.c | 4
> lxc-dave/drivers/usb/core/devio.c | 12 +-
> lxc-dave/drivers/usb/core/inode.c | 2
> lxc-dave/drivers/usb/core/usb.h | 2
> lxc-dave/files | 42 -----
> lxc-dave/fs/autofs/autofs_i.h | 2
> lxc-dave/fs/autofs/inode.c | 4
> lxc-dave/fs/autofs/root.c | 2
> lxc-dave/fs/compat.c | 4
> lxc-dave/fs/dnotify.c | 2
> lxc-dave/fs/exec.c | 8 -
> lxc-dave/fs/fcntl.c | 63 ++++++-----
> lxc-dave/fs/file_table.c | 2
> lxc-dave/fs/ioprio.c | 17 +--
> lxc-dave/fs/locks.c | 2
> lxc-dave/fs/ncpfs/inode.c | 20 +--
> lxc-dave/fs/proc/array.c | 2
> lxc-dave/fs/proc/base.c | 52 ++++++-----
> lxc-dave/fs/proc/inode.c | 4
> lxc-dave/fs/proc/internal.h | 6 -
> lxc-dave/fs/proc/root.c | 4
> lxc-dave/fs/proc/task_mmu.c | 4
> lxc-dave/fs/proc/task_nommu.c | 5 -
> lxc-dave/fs/smbfs/inode.c | 8 -
> lxc-dave/fs/smbfs/proc.c | 4
> lxc-dave/fs/smbfs/smbiod.c | 10 +-
> lxc-dave/include/linux/console_struct.h | 2
> lxc-dave/include/linux/fs.h | 8 +

```

> lxc-dave/include/linux/init_task.h | 22 ++--
> lxc-dave/include/linux/kernel.h | 4
> lxc-dave/include/linux/mmzone.h | 2
> lxc-dave/include/linux/n_r3964.h | 2
> lxc-dave/include/linux/ncp_mount.h | 2
> lxc-dave/include/linux/pid.h | 104 ++++++++-----
> lxc-dave/include/linux/proc_fs.h | 4
> lxc-dave/include/linux/sched.h | 42 ++++----
> lxc-dave/include/linux/smb_fs_sb.h | 2
> lxc-dave/include/linux/tty.h | 4
> lxc-dave/include/linux/vt_kern.h | 2
> lxc-dave/init/main.c | 2
> lxc-dave/ipc/mqueue.c | 8 -
> lxc-dave/kernel/capability.c | 8 -
> lxc-dave/kernel/cpuset.c | 10 +-
> lxc-dave/kernel/exit.c | 38 ++++----
> lxc-dave/kernel/fork.c | 23 ++--
> lxc-dave/kernel/futex.c | 2
> lxc-dave/kernel/pid.c | 152 ++++++++-----
> lxc-dave/kernel/rtmutex-debug.c | 1
> lxc-dave/kernel/signal.c | 26 ++---
> lxc-dave/kernel/sys.c | 28 ++---
> lxc-dave/kernel/sysctl.c | 10 +-
> lxc-dave/mm/mempolicy.c | 3
> lxc-dave/mm/migrate.c | 2
> 59 files changed, 453 insertions(+), 475 deletions(-)
>
> diff -puN include/linux/pid.h~rename-struct-pid include/linux/pid.h
> --- lxc/include/linux/pid.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/pid.h 2007-04-10 16:18:30.000000000 -0700
> @@ -3,23 +3,24 @@
>
> #include <linux/rcupdate.h>
>
> -enum pid_type
> +enum task_ref_type
> {
> - PIDTYPE_PID,
> - PIDTYPE_PGID,
> - PIDTYPE_SID,
> - PIDTYPE_MAX
> + REFTYPE_PID,
> + REFTYPE_PGID,
> + REFTYPE_SID,
> + REFTYPE_MAX
> };
>
> /*

```

```

> - * What is struct pid?
> + * What is struct task_ref?
> *
> - * A struct pid is the kernel's internal notion of a process identifier.
> - * It refers to individual tasks, process groups, and sessions. While
> - * there are processes attached to it the struct pid lives in a hash
> - * table, so it and then the processes that it refers to can be found
> - * quickly from the numeric pid value. The attached processes may be
> - * quickly accessed by following pointers from struct pid.
> + * A 'struct task_ref' is the kernel's internal notion of a process
> + * identifier. It refers to individual tasks, process groups, and
> + * sessions. While there are processes attached to it the
> + * 'struct task_ref' lives in a hash table, so it and then the
> + * processes that it refers to can be found quickly from the numeric
> + * pid value. The attached processes may be quickly accessed by
> + * following pointers from struct task_ref.
> *
> * Storing pid_t values in the kernel and referring to them later has a
> * problem. The process originally with that pid may have exited and the
> @@ -31,89 +32,98 @@ enum pid_type
> * the now useless task_struct is still kept. A task_struct plus a
> * stack consumes around 10K of low kernel memory. More precisely
> * this is THREAD_SIZE + sizeof(struct task_struct). By comparison
> - * a struct pid is about 64 bytes.
> + * a struct task_ref is about 64 bytes.
> *
> - * Holding a reference to struct pid solves both of these problems.
> + * Holding a reference to struct task_ref solves both of these problems.
> * It is small so holding a reference does not consume a lot of
> - * resources, and since a new struct pid is allocated when the numeric pid
> + * resources, and since a new struct task_ref is allocated when the numeric pid
> * value is reused (when pids wrap around) we don't mistakenly refer to new
> * processes.
> */
>
> -struct pid
> +struct task_ref
> {
>     atomic_t count;
>     /* Try to keep pid_chain in the same cacheline as nr for find_pid */
>     - int nr;
>     + int pid;
>     struct hlist_node pid_chain;
>     - /* lists of tasks that use this pid */
>     - struct hlist_head tasks[PIDTYPE_MAX];
>     + /*
>     +  * lists of tasks that use this pid.
>     +  * For instance, ->tasks[REFTYPE_SID]

```

```

> + * has all tasks with a session id of
> + * the number in ->pid.
> + */
> + struct hlist_head tasks[REFTYPE_MAX];
>   struct rcu_head rcu;
> };
>
> -extern struct pid init_struct_pid;
> +extern struct task_ref init_task_ref;
>
> struct pid_link
> {
>   struct hlist_node node;
> - struct pid *pid;
> + struct task_ref *tref;
> };
>
> -static inline struct pid *get_pid(struct pid *pid)
> +static inline struct task_ref *get_pid(struct task_ref *tref)
> {
> - if (pid)
> -   atomic_inc(&pid->count);
> - return pid;
> + if (tref)
> +   atomic_inc(&tref->count);
> + return tref;
> }
>
> -extern void FASTCALL(put_pid(struct pid *pid));
> -extern struct task_struct *FASTCALL(pid_task(struct pid *pid, enum pid_type));
> -extern struct task_struct *FASTCALL(get_pid_task(struct pid *pid,
> -   enum pid_type));
> +extern void FASTCALL(put_task_ref(struct task_ref *tref));
> +extern struct task_struct *FASTCALL(pid_task(struct task_ref *tref,
> +   enum task_ref_type));
> +extern struct task_struct *FASTCALL(get_pid_task(struct task_ref *tref,
> +   enum task_ref_type));
>
> -extern struct pid *get_task_pid(struct task_struct *task, enum pid_type type);
> +extern struct task_ref *get_task_ref(struct task_struct *task,
> +   enum task_ref_type type);
>
> /*
>  * attach_pid() and detach_pid() must be called with the tasklist_lock
>  * write-held.
>  */
> -extern int FASTCALL(attach_pid(struct task_struct *task,
> -   enum pid_type type, struct pid *pid));

```

```

> -extern void FASTCALL(detach_pid(struct task_struct *task, enum pid_type));
> -extern void FASTCALL(transfer_pid(struct task_struct *old,
> -    struct task_struct *new, enum pid_type));
> +extern int FASTCALL(attach_task_ref(struct task_struct *task,
> +    enum task_ref_type type,
> +    struct task_ref *tref));
> +extern void FASTCALL(detach_task_ref(struct task_struct *task, enum task_ref_type));
> +extern void FASTCALL(transfer_task_ref(struct task_struct *old,
> +    struct task_struct *new,
> +    enum task_ref_type));
>
> /*
>  * look up a PID in the hash table. Must be called with the tasklist_lock
>  * or rcu_read_lock() held.
>  */
> -extern struct pid *FASTCALL(find_pid(int nr));
> +extern struct task_ref *FASTCALL(find_task(int nr));
>
> /*
>  * Lookup a PID in the hash table, and return with it's count elevated.
>  */
> -extern struct pid *find_get_pid(int nr);
> -extern struct pid *find_ge_pid(int nr);
> +extern struct task_ref *find_get_pid(int nr);
> +extern struct task_ref *find_ge_pid(int nr);
>
> -extern struct pid *alloc_pid(void);
> -extern void FASTCALL(free_pid(struct pid *pid));
> +extern struct task_ref *alloc_task_ref(void);
> +extern void FASTCALL(free_task_ref(struct task_ref *tref));
>
> -static inline pid_t pid_nr(struct pid *pid)
> +static inline pid_t tref_to_pid(struct task_ref *tref)
> {
>     pid_t nr = 0;
>     - if (pid)
>     - nr = pid->nr;
>     + if (tref)
>     + nr = tref->pid;
>     return nr;
> }
>
> -#define do_each_pid_task(pid, type, task) \
> +#define do_each_referenced_task(tref, type, task) \
> do { \
>     struct hlist_node *pos___; \
>     - if (pid != NULL) \
>     + if (tref != NULL) \

```

```

> hlist_for_each_entry_rcu((task), pos____, \
> - &pid->tasks[type], pids[type].node) {
> + &tref->tasks[type], pids[type].node) {
>
> -#define while_each_pid_task(pid, type, task) \
> +#define while_each_referenced_task(tref, type, task) \
> } \
> } while (0)
>
> diff -puN include/linux/console_struct.h~rename-struct-pid include/linux/console_struct.h
> --- lxc/include/linux/console_struct.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/console_struct.h 2007-04-10 16:18:30.000000000 -0700
> @@ -56,7 +56,7 @@ struct vc_data {
> struct tty_struct *vc_tty; /* TTY we are attached to */
> /* data for manual vt switching */
> struct vt_mode vt_mode;
> - struct pid *vt_pid;
> + struct task_ref *vt_tref;
> int vt_newvt;
> wait_queue_head_t paste_wait;
> /* mode flags */
> diff -puN include/linux/fs.h~rename-struct-pid include/linux/fs.h
> --- lxc/include/linux/fs.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/fs.h 2007-04-10 16:18:30.000000000 -0700
> @@ -687,8 +687,9 @@ extern struct block_device *l_BDEV(struct
>
> struct fown_struct {
> rwlock_t lock; /* protects pid, uid, euid fields */
> - struct pid *pid; /* pid or -pgrp where SIGIO should be sent */
> - enum pid_type pid_type; /* Kind of process group SIGIO should be sent to */
> + struct task_ref *tref; /* pid or -pgrp where SIGIO should be sent */
> + enum task_ref_type task_ref_type;
> + /* Kind of process group SIGIO should be sent to */
> uid_t uid, euid; /* uid/euid of process setting the owner */
> int signum; /* posix.1b rt signal to be delivered on IO */
> };
> @@ -925,7 +926,8 @@ extern void kill_fasync(struct fasync_st
> /* only for net: no internal synchronization */
> extern void __kill_fasync(struct fasync_struct *, int, int);
>
> -extern int __f_setown(struct file *filp, struct pid *, enum pid_type, int force);
> +extern int __f_setown(struct file *filp, struct task_ref *,
> + enum task_ref_type, int force);
> extern int f_setown(struct file *filp, unsigned long arg, int force);
> extern void f_delown(struct file *filp);
> extern pid_t f_getown(struct file *filp);
> diff -puN include/linux/init_task.h~rename-struct-pid include/linux/init_task.h
> --- lxc/include/linux/init_task.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700

```

```

> +++ lxc-dave/include/linux/init_task.h 2007-04-10 16:18:30.000000000 -0700
> @@ -89,15 +89,15 @@ extern struct nsproxy init_nsproxy;
>
> extern struct group_info init_groups;
>
> #define INIT_STRUCT_PID { \
> +#define INIT_TASK_REF { \
> .count = ATOMIC_INIT(1), \
> - .nr = 0, \
> - /* Don't put this struct pid in pid_hash */ \
> + .pid = 0, \
> + /* Don't put this task_ref in pid_hash */ \
> .pid_chain = { .next = NULL, .pprev = NULL }, \
> .tasks = { \
> - { .first = &init_task.pids[PIDTYPE_PID].node }, \
> - { .first = &init_task.pids[PIDTYPE_PGID].node }, \
> - { .first = &init_task.pids[PIDTYPE_SID].node }, \
> + { .first = &init_task.pids[REFTYPE_PID].node }, \
> + { .first = &init_task.pids[REFTYPE_PGID].node }, \
> + { .first = &init_task.pids[REFTYPE_SID].node }, \
> }, \
> .rcu = RCU_HEAD_INIT, \
> }
> @@ -106,9 +106,9 @@ extern struct group_info init_groups;
> { \
> .node = { \
> .next = NULL, \
> - .pprev = &init_struct_pid.tasks[type].first, \
> + .pprev = &init_task_ref.tasks[type].first, \
> }, \
> - .pid = &init_struct_pid, \
> + .tref = &init_task_ref, \
> }
>
> /*
> @@ -162,9 +162,9 @@ extern struct group_info init_groups;
> .fs_excl = ATOMIC_INIT(0), \
> .pi_lock = SPIN_LOCK_UNLOCKED, \
> .pids = { \
> - [PIDTYPE_PID] = INIT_PID_LINK(PIDTYPE_PID), \
> - [PIDTYPE_PGID] = INIT_PID_LINK(PIDTYPE_PGID), \
> - [PIDTYPE_SID] = INIT_PID_LINK(PIDTYPE_SID), \
> + [REFTYPE_PID] = INIT_PID_LINK(REFTYPE_PID), \
> + [REFTYPE_PGID] = INIT_PID_LINK(REFTYPE_PGID), \
> + [REFTYPE_SID] = INIT_PID_LINK(REFTYPE_SID), \
> }, \
> INIT_TRACE_IRQFLAGS \
> INIT_LOCKDEP \

```



```

> diff -puN include/linux/kernel.h~rename-struct-pid include/linux/kernel.h
> --- lxc/include/linux/kernel.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/kernel.h 2007-04-10 16:18:30.000000000 -0700
> @@ -135,8 +135,8 @@ extern unsigned long long memparse(char
> extern int core_kernel_text(unsigned long addr);
> extern int __kernel_text_address(unsigned long addr);
> extern int kernel_text_address(unsigned long addr);
> -struct pid;
> -extern struct pid *session_of_pgrp(struct pid *pgrp);
> +struct tref;
> +extern struct task_ref *session_of_pgrp(struct task_ref *pgrp);
>
> extern void dump_thread(struct pt_regs *regs, struct user *dump);
>
> diff -puN include/linux/mmzone.h~rename-struct-pid include/linux/mmzone.h
> --- lxc/include/linux/mmzone.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/mmzone.h 2007-04-10 16:18:30.000000000 -0700
> @@ -619,7 +619,7 @@ int sysctl_min_slab_ratio_sysctl_handler
> #ifndef CONFIG_NEED_MULTIPLE_NODES
>
> extern struct pglist_data contig_page_data;
> -#define NODE_DATA(nid) (&contig_page_data)
> +#define NODE_DATA(nid) ({ (void)nid; &contig_page_data; })
> #define NODE_MEM_MAP(nid) mem_map
> #define MAX_NODES_SHIFT 1
>
> diff -puN include/linux/ncp_mount.h~rename-struct-pid include/linux/ncp_mount.h
> --- lxc/include/linux/ncp_mount.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/ncp_mount.h 2007-04-10 16:18:30.000000000 -0700
> @@ -75,7 +75,7 @@ struct ncp_mount_data_kernel {
> unsigned int int_flags; /* internal flags */
> #define NCP_IMOUNT_LOGGEDIN_POSSIBLE 0x0001
> __kernel_uid32_t mounted_uid; /* Who may umount() this filesystem? */
> - struct pid *wdog_pid; /* Who cares for our watchdog packets? */
> + struct task_ref *wdog_tref; /* Who cares for our watchdog packets? */
> unsigned int ncp_fd; /* The socket to the ncp port */
> unsigned int time_out; /* How long should I wait after
> sending a NCP request? */
> diff -puN include/linux/n_r3964.h~rename-struct-pid include/linux/n_r3964.h
> --- lxc/include/linux/n_r3964.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/n_r3964.h 2007-04-10 16:18:30.000000000 -0700
> @@ -116,7 +116,7 @@ struct r3964_message;
>
> struct r3964_client_info {
> spinlock_t lock;
> - struct pid *pid;
> + struct task_ref *tref;
> unsigned int sig_flags;

```

```

>
> struct r3964_client_info *next;
> diff -puN include/linux/proc_fs.h~rename-struct-pid include/linux/proc_fs.h
> --- lxc/include/linux/proc_fs.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/proc_fs.h 2007-04-10 16:18:30.000000000 -0700
> @@ -265,7 +265,7 @@ union proc_op {
> };
>
> struct proc_inode {
> - struct pid *pid;
> + struct task_ref *tref;
> int fd;
> union proc_op op;
> struct proc_dir_entry *pde;
> @@ -283,7 +283,7 @@ static inline struct proc_dir_entry *PDE
> }
>
> struct proc_maps_private {
> - struct pid *pid;
> + struct task_ref *tref;
> struct task_struct *task;
> #ifdef CONFIG_MMU
> struct vm_area_struct *tail_vma;
> diff -puN include/linux/sched.h~rename-struct-pid include/linux/sched.h
> --- lxc/include/linux/sched.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/sched.h 2007-04-10 16:18:30.000000000 -0700
> @@ -449,7 +449,7 @@ struct signal_struct {
>
> /* job control IDs */
> pid_t pgrp;
> - struct pid *tty_old_pgrp;
> + struct task_ref *tty_old_pgrp;
>
> union {
> pid_t session __deprecated;
> @@ -906,7 +906,7 @@ struct task_struct {
> struct task_struct *group_leader; /* threadgroup leader */
>
> /* PID/PID hash table linkage. */
> - struct pid_link pids[PIDTYPE_MAX];
> + struct pid_link pids[REFTYPE_MAX];
> struct list_head thread_group;
>
> struct completion *vfork_done; /* for vfork() */
> @@ -1116,24 +1116,24 @@ static inline void set_signal_session(st
> sig->__session = session;
> }
>
>

```

```

> -static inline struct pid *task_pid(struct task_struct *task)
> +static inline struct task_ref *task_pid(struct task_struct *task)
> {
> - return task->pids[PIDTYPE_PID].pid;
> + return task->pids[REFTYPE_PID].tref;
> }
>
> -static inline struct pid *task_tgid(struct task_struct *task)
> +static inline struct task_ref *task_tgid(struct task_struct *task)
> {
> - return task->group_leader->pids[PIDTYPE_PID].pid;
> + return task->group_leader->pids[REFTYPE_PID].tref;
> }
>
> -static inline struct pid *task_pgrp(struct task_struct *task)
> +static inline struct task_ref *task_pgrp(struct task_struct *task)
> {
> - return task->group_leader->pids[PIDTYPE_PGID].pid;
> + return task->group_leader->pids[REFTYPE_PGID].tref;
> }
>
> -static inline struct pid *task_session(struct task_struct *task)
> +static inline struct task_ref *task_session(struct task_struct *task)
> {
> - return task->group_leader->pids[PIDTYPE_SID].pid;
> + return task->group_leader->pids[REFTYPE_SID].tref;
> }
>
> /**
> @@ -1146,7 +1146,7 @@ static inline struct pid *task_session(s
> */
> static inline int pid_alive(struct task_struct *p)
> {
> - return p->pids[PIDTYPE_PID].pid != NULL;
> + return p->pids[REFTYPE_PID].tref != NULL;
> }
>
> /**
> @@ -1160,7 +1160,7 @@ static inline int is_init(struct task_st
> return tsk->pid == 1;
> }
>
> -extern struct pid *cad_pid;
> +extern struct task_ref *cad_tref;
>
> extern void free_task(struct task_struct *tsk);
> #define get_task_struct(tsk) do { atomic_inc(&(tsk)->usage); } while(0)
> @@ -1307,8 +1307,8 @@ extern struct task_struct init_task;

```

```

>
> extern struct mm_struct init_mm;
>
> #define find_task_by_pid(nr) find_task_by_pid_type(PIDTYPE_PID, nr)
> -extern struct task_struct *find_task_by_pid_type(int type, int pid);
> +#define find_task_by_pid(nr) find_task_by_ref_type(REFTYPE_PID, nr)
> +extern struct task_struct *find_task_by_ref_type(int type, int pid);
> extern void __set_special_pids(pid_t session, pid_t pgrp);
>
> /* per-UID process charging. */
> @@ -1365,12 +1365,12 @@ extern int send_sig_info(int, struct sig
> extern int send_group_sig_info(int, struct siginfo *, struct task_struct *);
> extern int force_sigsegv(int, struct task_struct *);
> extern int force_sig_info(int, struct siginfo *, struct task_struct *);
> -extern int __kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
> -extern int kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
> -extern int kill_pid_info(int sig, struct siginfo *info, struct pid *pid);
> -extern int kill_pid_info_as_uid(int, struct siginfo *, struct pid *, uid_t, uid_t, u32);
> -extern int kill_pgrp(struct pid *pid, int sig, int priv);
> -extern int kill_pid(struct pid *pid, int sig, int priv);
> +extern int __kill_pgrp_info(int sig, struct siginfo *info, struct task_ref *pgrp);
> +extern int kill_pgrp_info(int sig, struct siginfo *info, struct task_ref *pgrp);
> +extern int kill_pid_info(int sig, struct siginfo *info, struct task_ref *pid);
> +extern int kill_pid_info_as_uid(int, struct siginfo *, struct task_ref *, uid_t, uid_t, u32);
> +extern int kill_pgrp(struct task_ref *tref, int sig, int priv);
> +extern int kill_pid(struct task_ref *tref, int sig, int priv);
> extern int kill_proc_info(int, struct siginfo *, pid_t);
> extern void do_notify_parent(struct task_struct *, int);
> extern void do_notify_parent_cldstop(struct task_struct *, int);
> @@ -1388,7 +1388,7 @@ extern int do_sigaltstack(const stack_t
>
> static inline int kill_cad_pid(int sig, int priv)
> {
> - return kill_pid(cad_pid, sig, priv);
> + return kill_pid(cad_tref, sig, priv);
> }
>
> /* These can be the second arg to send_sig_info/send_group_sig_info. */
> diff -puN include/linux/smb_fs_sb.h~rename-struct-pid include/linux/smb_fs_sb.h
> --- lxc/include/linux/smb_fs_sb.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/smb_fs_sb.h 2007-04-10 16:18:30.000000000 -0700
> @@ -55,7 +55,7 @@ struct smb_sb_info {
>  * generation is incremented.
>  */
>  unsigned int generation;
>  - struct pid *conn_pid;
>  + struct task_ref *conn_tref;
>  struct smb_conn_opt opt;

```

```

> wait_queue_head_t conn_wq;
> int conn_complete;
> diff -puN include/linux/tty.h~rename-struct-pid include/linux/tty.h
> --- lxc/include/linux/tty.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/tty.h 2007-04-10 16:18:30.000000000 -0700
> @@ -197,8 +197,8 @@ struct tty_struct {
> struct mutex termios_mutex;
> struct ktermios *termios, *termios_locked;
> char name[64];
> - struct pid *pgrp;
> - struct pid *session;
> + struct task_ref *pgrp;
> + struct task_ref *session;
> unsigned long flags;
> int count;
> struct winsize winsize;
> diff -puN include/linux/vt_kern.h~rename-struct-pid include/linux/vt_kern.h
> --- lxc/include/linux/vt_kern.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/include/linux/vt_kern.h 2007-04-10 16:18:30.000000000 -0700
> @@ -87,7 +87,7 @@ extern char vt_dont_switch;
>
> struct vt_spawn_console {
> spinlock_t lock;
> - struct pid *pid;
> + struct task_ref *tref;
> int sig;
> };
> extern struct vt_spawn_console vt_spawn_con;
> diff -puN drivers/char/keyboard.c~rename-struct-pid drivers/char/keyboard.c
> --- lxc/drivers/char/keyboard.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/char/keyboard.c 2007-04-10 16:18:30.000000000 -0700
> @@ -110,7 +110,7 @@ static struct kbd_struct *kbd = kbd_tabl
>
> struct vt_spawn_console vt_spawn_con = {
> .lock = SPIN_LOCK_UNLOCKED,
> - .pid = NULL,
> + .tref = NULL,
> .sig = 0,
> };
>
> @@ -559,10 +559,10 @@ static void fn_compose(struct vc_data *v
> static void fn_spawn_con(struct vc_data *vc)
> {
> spin_lock(&vt_spawn_con.lock);
> - if (vt_spawn_con.pid)
> - if (kill_pid(vt_spawn_con.pid, vt_spawn_con.sig, 1)) {
> - put_pid(vt_spawn_con.pid);
> - vt_spawn_con.pid = NULL;

```

```

> + if (vt_spawn_con.tref)
> + if (kill_pid(vt_spawn_con.tref, vt_spawn_con.sig, 1)) {
> + put_task_ref(vt_spawn_con.tref);
> + vt_spawn_con.tref = NULL;
> }
> spin_unlock(&vt_spawn_con.lock);
> }
> diff -puN drivers/char/n_r3964.c~rename-struct-pid drivers/char/n_r3964.c
> --- lxc/drivers/char/n_r3964.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/char/n_r3964.c 2007-04-10 16:18:30.000000000 -0700
> @@ -119,8 +119,8 @@ static void transmit_block(struct r3964_
> static void receive_char(struct r3964_info *pInfo, const unsigned char c);
> static void receive_error(struct r3964_info *pInfo, const char flag);
> static void on_timeout(unsigned long priv);
> -static int enable_signals(struct r3964_info *pInfo, struct pid *pid, int arg);
> -static int read_telegram(struct r3964_info *pInfo, struct pid *pid,
> +static int enable_signals(struct r3964_info *pInfo, struct task_ref *tref, int arg);
> +static int read_telegram(struct r3964_info *pInfo, struct task_ref *tref,
> unsigned char __user * buf);
> static void add_msg(struct r3964_client_info *pClient, int msg_id, int arg,
> int error_code, struct r3964_block_header *pBlock);
> @@ -745,19 +745,19 @@ static void on_timeout(unsigned long pri
> }
>
> static struct r3964_client_info *findClient(struct r3964_info *pInfo,
> - struct pid *pid)
> + struct task_ref *tref)
> {
> struct r3964_client_info *pClient;
>
> for (pClient = pInfo->firstClient; pClient; pClient = pClient->next) {
> - if (pClient->pid == pid) {
> + if (pClient->tref == tref) {
> return pClient;
> }
> }
> return NULL;
> }
>
> -static int enable_signals(struct r3964_info *pInfo, struct pid *pid, int arg)
> +static int enable_signals(struct r3964_info *pInfo, struct task_ref *tref, int arg)
> {
> struct r3964_client_info *pClient;
> struct r3964_client_info **ppClient;
> @@ -769,9 +769,9 @@ static int enable_signals(struct r3964_i
> ppClient = &(*ppClient)->next) {
> pClient = *ppClient;
>

```

```

> - if (pClient->pid == pid) {
> + if (pClient->tref == tref) {
>   TRACE_PS("removing client %d from client list",
> -     pid_nr(pid));
> +     tref_to_pid(tref));
>   *ppClient = pClient->next;
>   while (pClient->msg_count) {
>     pMsg = remove_msg(pInfo, pClient);
> @@ -781,7 +781,7 @@ static int enable_signals(struct r3964_i
>     "kfree %p", pMsg);
>   }
> }
> - put_pid(pClient->pid);
> + put_task_ref(pClient->tref);
>   kfree(pClient);
>   TRACE_M("enable_signals - kfree %p", pClient);
>   return 0;
> @@ -789,7 +789,7 @@ static int enable_signals(struct r3964_i
> }
> return -EINVAL;
> } else {
> - pClient = findClient(pInfo, pid);
> + pClient = findClient(pInfo, tref);
>   if (pClient) {
>     /* update signal options */
>     pClient->sig_flags = arg;
> @@ -801,10 +801,10 @@ static int enable_signals(struct r3964_i
>     if (pClient == NULL)
>       return -ENOMEM;
>
> - TRACE_PS("add client %d to client list", pid_nr(pid));
> + TRACE_PS("add client %d to client list", tref_to_pid(tref));
>   spin_lock_init(&pClient->lock);
>   pClient->sig_flags = arg;
> - pClient->pid = get_pid(pid);
> + pClient->tref = get_pid(tref);
>   pClient->next = pInfo->firstClient;
>   pClient->first_msg = NULL;
>   pClient->last_msg = NULL;
> @@ -817,7 +817,7 @@ static int enable_signals(struct r3964_i
>   return 0;
> }
>
> -static int read_telegram(struct r3964_info *pInfo, struct pid *pid,
> +static int read_telegram(struct r3964_info *pInfo, struct task_ref *tref,
>   unsigned char __user * buf)
> {
>   struct r3964_client_info *pClient;

```

```

> @@ -827,7 +827,7 @@ static int read_telegram(struct r3964_in
>     return -EINVAL;
> }
>
> - pClient = findClient(plInfo, pid);
> + pClient = findClient(plInfo, tref);
> if (pClient == NULL) {
>     return -EINVAL;
> }
> @@ -899,7 +899,7 @@ queue_the_message:
> }
> /* Send SIGIO signal to client process: */
> if (pClient->sig_flags & R3964_USE_SIGIO) {
> - kill_pid(pClient->pid, SIGIO, 1);
> + kill_pid(pClient->tref, SIGIO, 1);
> }
> }
>
> @@ -933,7 +933,7 @@ static void remove_client_block(struct r
> {
>     struct r3964_block_header *block;
>
> - TRACE_PS("remove_client_block PID %d", pid_nr(pClient->pid));
> + TRACE_PS("remove_client_block PID %d", tref_to_pid(pClient->tref));
>
>     block = pClient->next_block_to_read;
>     if (block) {
> @@ -1037,7 +1037,7 @@ static void r3964_close(struct tty_struc
>     TRACE_M("r3964_close - msg kfree %p", pMsg);
> }
> }
> - put_pid(pClient->pid);
> + put_task_ref(pClient->tref);
> kfree(pClient);
> TRACE_M("r3964_close - client kfree %p", pClient);
> pClient = pNext;
> diff -puN drivers/char/tty_io.c~rename-struct-pid drivers/char/tty_io.c
> --- lxc/drivers/char/tty_io.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/char/tty_io.c 2007-04-10 16:18:30.000000000 -0700
> @@ -1355,7 +1355,7 @@ static void do_tty_hangup(struct work_st
>
>     read_lock(&tasklist_lock);
>     if (tty->session) {
> - do_each_pid_task(tty->session, PIDTYPE_SID, p) {
> + do_each_referenced_task(tty->session, REFTYPE_SID, p) {
>         spin_lock_irq(&p->sigband->siglock);
>         if (p->signal->tty == tty)
>             p->signal->tty = NULL;

```



```

> @@ -1365,17 +1365,17 @@ static void do_tty_hangup(struct work_st
> }
> __group_send_sig_info(SIGHUP, SEND_SIG_PRIV, p);
> __group_send_sig_info(SIGCONT, SEND_SIG_PRIV, p);
> - put_pid(p->signal->tty_old_pgrp); /* A noop */
> + put_task_ref(p->signal->tty_old_pgrp); /* A noop */
> if (tty->pgrp)
> p->signal->tty_old_pgrp = get_pid(tty->pgrp);
> spin_unlock_irq(&p->sigband->siglock);
> - } while_each_pid_task(tty->session, PIDTYPE_SID, p);
> + } while_each_referenced_task(tty->session, REFTYPE_SID, p);
> }
> read_unlock(&tasklist_lock);
>
> tty->flags = 0;
> - put_pid(tty->session);
> - put_pid(tty->pgrp);
> + put_task_ref(tty->session);
> + put_task_ref(tty->pgrp);
> tty->session = NULL;
> tty->pgrp = NULL;
> tty->ctrl_status = 0;
> @@ -1462,12 +1462,12 @@ int tty_hung_up_p(struct file * filp)
>
> EXPORT_SYMBOL(tty_hung_up_p);
>
> -static void session_clear_tty(struct pid *session)
> +static void session_clear_tty(struct task_ref *session)
> {
> struct task_struct *p;
> - do_each_pid_task(session, PIDTYPE_SID, p) {
> + do_each_referenced_task(session, REFTYPE_SID, p) {
> proc_clear_tty(p);
> - } while_each_pid_task(session, PIDTYPE_SID, p);
> + } while_each_referenced_task(session, REFTYPE_SID, p);
> }
>
> /**
> @@ -1497,7 +1497,7 @@ static void session_clear_tty(struct pid
> void disassociate_ctty(int on_exit)
> {
> struct tty_struct *tty;
> - struct pid *tty_pgrp = NULL;
> + struct task_ref *tty_pgrp = NULL;
>
> lock_kernel();
>
> @@ -1510,7 +1510,7 @@ void disassociate_ctty(int on_exit)

```

```

> if (on_exit && tty->driver->type != TTY_DRIVER_TYPE_PTY)
>   tty_vhangup(tty);
> } else if (on_exit) {
> - struct pid *old_pgrp;
> + struct task_ref *old_pgrp;
>   spin_lock_irq(&current->sigband->siglock);
>   old_pgrp = current->signal->tty_old_pgrp;
>   current->signal->tty_old_pgrp = NULL;
> @@ -1518,7 +1518,7 @@ void disassociate_ctty(int on_exit)
>   if (old_pgrp) {
>     kill_pgrp(old_pgrp, SIGHUP, on_exit);
>     kill_pgrp(old_pgrp, SIGCONT, on_exit);
> -   put_pid(old_pgrp);
> +   put_task_ref(old_pgrp);
>   }
>   mutex_unlock(&tty_mutex);
>   unlock_kernel();
> @@ -1528,11 +1528,11 @@ void disassociate_ctty(int on_exit)
>   kill_pgrp(tty_pgrp, SIGHUP, on_exit);
>   if (!on_exit)
>     kill_pgrp(tty_pgrp, SIGCONT, on_exit);
> -   put_pid(tty_pgrp);
> +   put_task_ref(tty_pgrp);
> }
>
>   spin_lock_irq(&current->sigband->siglock);
> -   put_pid(current->signal->tty_old_pgrp);
> +   put_task_ref(current->signal->tty_old_pgrp);
>   current->signal->tty_old_pgrp = NULL;
>   spin_unlock_irq(&current->sigband->siglock);
>
> @@ -1540,8 +1540,8 @@ void disassociate_ctty(int on_exit)
>   /* It is possible that do_tty_hangup has free'd this tty */
>   tty = get_current_tty();
>   if (tty) {
> -   put_pid(tty->session);
> -   put_pid(tty->pgrp);
> +   put_task_ref(tty->session);
> +   put_task_ref(tty->pgrp);
>   tty->session = NULL;
>   tty->pgrp = NULL;
>   } else {
> @@ -2757,18 +2757,18 @@ static int tty_fasync(int fd, struct fil
>   return retval;
>
>   if (on) {
> -   enum pid_type type;
> -   struct pid *pid;

```

```

> + enum task_ref_type type;
> + struct task_ref *tref;
> if (!waitqueue_active(&tty->read_wait))
>   tty->minimum_to_wake = 1;
> if (tty->pgrp) {
> - pid = tty->pgrp;
> - type = PIDTYPE_PGID;
> + tref = tty->pgrp;
> + type = REFTYPE_PGID;
> } else {
> - pid = task_pid(current);
> - type = PIDTYPE_PID;
> + tref = task_pid(current);
> + type = REFTYPE_PID;
> }
> - retval = __f_setown(filp, pid, type, 0);
> + retval = __f_setown(filp, tref, type, 0);
> if (retval)
>   return retval;
> } else {
> @@ -3014,7 +3014,7 @@ static int tiocgpgrp(struct tty_struct *
> */
> if (tty == real_tty && current->signal->tty != real_tty)
>   return -ENOTTY;
> - return put_user(pid_nr(real_tty->pgrp), p);
> + return put_user(tref_to_pid(real_tty->pgrp), p);
> }
>
> /**
> @@ -3031,7 +3031,7 @@ static int tiocgpgrp(struct tty_struct *
>
> static int tiocspgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t __user *p)
> {
> - struct pid *pgrp;
> + struct task_ref *pgrp;
>   pid_t pgrp_nr;
>   int retval = tty_check_change(real_tty);
>
> @@ -3048,7 +3048,7 @@ static int tiocspgrp(struct tty_struct *
>   if (pgrp_nr < 0)
>     return -EINVAL;
>   rcu_read_lock();
> - pgrp = find_pid(pgrp_nr);
> + pgrp = find_task(pgrp_nr);
>   retval = -ESRCH;
>   if (!pgrp)
>     goto out_unlock;
> @@ -3056,7 +3056,7 @@ static int tiocspgrp(struct tty_struct *

```

```

> if (session_of_pgrp(pgrp) != task_session(current))
> goto out_unlock;
> retval = 0;
> - put_pid(real_tty->pgrp);
> + put_task_ref(real_tty->pgrp);
> real_tty->pgrp = get_pid(pgrp);
> out_unlock:
> rcu_read_unlock();
> @@ -3085,7 +3085,7 @@ static int tiocgsid(struct tty_struct *t
> return -ENOTTY;
> if (!real_tty->session)
> return -ENOTTY;
> - return put_user(pid_nr(real_tty->session), p);
> + return put_user(tref_to_pid(real_tty->session), p);
> }
>
> /**
> @@ -3383,7 +3383,7 @@ void __do_SAK(struct tty_struct *tty)
> tty_hangup(tty);
> #else
> struct task_struct *g, *p;
> - struct pid *session;
> + struct task_ref *session;
> int i;
> struct file *filp;
> struct fdtable *fdt;
> @@ -3399,12 +3399,12 @@ void __do_SAK(struct tty_struct *tty)
>
> read_lock(&tasklist_lock);
> /* Kill the entire session */
> - do_each_pid_task(session, PIDTYPE_SID, p) {
> + do_each_referenced_task(session, REFTYPE_SID, p) {
> printk(KERN_NOTICE "SAK: killed process %d"
> " (%s): process_session(p)==tty->session\n",
> p->pid, p->comm);
> send_sig(SIGKILL, p, 1);
> - } while_each_pid_task(session, PIDTYPE_SID, p);
> + } while_each_referenced_task(session, REFTYPE_SID, p);
> /* Now kill any processes that happen to have the
> * tty open.
> */
> @@ -3849,12 +3849,12 @@ static void __proc_set_tty(struct task_s
> {
> if (tty) {
> /* We should not have a session or pgrp to here but.... */
> - put_pid(tty->session);
> - put_pid(tty->pgrp);
> + put_task_ref(tty->session);

```

```

> + put_task_ref(tty->pgrp);
> tty->session = get_pid(task_session(tsk));
> tty->pgrp = get_pid(task_pgrp(tsk));
> }
> - put_pid(tsk->signal->tty_old_pgrp);
> + put_task_ref(tsk->signal->tty_old_pgrp);
> tsk->signal->tty = tty;
> tsk->signal->tty_old_pgrp = NULL;
> }
> diff -puN drivers/char/vt.c~rename-struct-pid drivers/char/vt.c
> --- lxc/drivers/char/vt.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/char/vt.c 2007-04-10 16:18:30.000000000 -0700
> @@ -903,7 +903,7 @@ void vc_deallocate(unsigned int currcons
> if (vc_cons_allocated(currcons)) {
> struct vc_data *vc = vc_cons[currcons].d;
> vc->vc_sw->con_deinit(vc);
> - put_pid(vc->vt_pid);
> + put_task_ref(vc->vt_tref);
> module_put(vc->vc_sw->owner);
> if (vc->vc_kmalloced)
> kfree(vc->vc_screenbuf);
> diff -puN drivers/char/vt_ioctl.c~rename-struct-pid drivers/char/vt_ioctl.c
> --- lxc/drivers/char/vt_ioctl.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/char/vt_ioctl.c 2007-04-10 16:18:30.000000000 -0700
> @@ -651,8 +651,8 @@ int vt_ioctl(struct tty_struct *tty, str
> return -EINVAL;
>
> spin_lock_irq(&vt_spawn_con.lock);
> - put_pid(vt_spawn_con.pid);
> - vt_spawn_con.pid = get_pid(task_pid(current));
> + put_task_ref(vt_spawn_con.tref);
> + vt_spawn_con.tref = get_pid(task_pid(current));
> vt_spawn_con.sig = arg;
> spin_unlock_irq(&vt_spawn_con.lock);
> return 0;
> @@ -672,8 +672,8 @@ int vt_ioctl(struct tty_struct *tty, str
> vc->vt_mode = tmp;
> /* the frsig is ignored, so we set it to 0 */
> vc->vt_mode.frsig = 0;
> - put_pid(vc->vt_pid);
> - vc->vt_pid = get_pid(task_pid(current));
> + put_task_ref(vc->vt_tref);
> + vc->vt_tref = get_pid(task_pid(current));
> /* no switch is required -- saw@shade.msu.ru */
> vc->vt_newvt = -1;
> release_console_sem();
> @@ -1076,8 +1076,8 @@ void reset_vc(struct vc_data *vc)
> vc->vt_mode.relsig = 0;

```

```

> vc->vt_mode.acqsig = 0;
> vc->vt_mode.frsig = 0;
> - put_pid(vc->vt_pid);
> - vc->vt_pid = NULL;
> + put_task_ref(vc->vt_tref);
> + vc->vt_tref = NULL;
> vc->vt_newvt = -1;
> if (!in_interrupt()) /* Via keyboard.c:SAK() - akpm */
> reset_palette(vc);
> @@ -1150,7 +1150,7 @@ static void complete_change_console(stru
> * tell us if the process has gone or something else
> * is awry
> */
> - if (kill_pid(vc->vt_pid, vc->vt_mode.acqsig, 1) != 0) {
> + if (kill_pid(vc->vt_tref, vc->vt_mode.acqsig, 1) != 0) {
> /*
> * The controlling process has died, so we revert back to
> * normal operation. In this case, we'll also change back
> @@ -1210,7 +1210,7 @@ void change_console(struct vc_data *new_
> * tell us if the process has gone or something else
> * is awry
> */
> - if (kill_pid(vc->vt_pid, vc->vt_mode.relsig, 1) == 0) {
> + if (kill_pid(vc->vt_tref, vc->vt_mode.relsig, 1) == 0) {
> /*
> * It worked. Mark the vt to switch to and
> * return. The process needs to send us a
> diff -puN drivers/net/tun.c~rename-struct-pid drivers/net/tun.c
> --- lxc/drivers/net/tun.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/net/tun.c 2007-04-10 16:18:30.000000000 -0700
> @@ -713,7 +713,7 @@ static int tun_chr_fasync(int fd, struct
> return ret;
>
> if (on) {
> - ret = __f_setown(file, task_pid(current), PIDTYPE_PID, 0);
> + ret = __f_setown(file, task_pid(current), REFTYPE_PID, 0);
> if (ret)
> return ret;
> tun->flags |= TUN_FASYNC;
> diff -puN drivers/s390/char/fs3270.c~rename-struct-pid drivers/s390/char/fs3270.c
> --- lxc/drivers/s390/char/fs3270.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/s390/char/fs3270.c 2007-04-10 16:18:30.000000000 -0700
> @@ -27,7 +27,7 @@ static struct raw3270_fn fs3270_fn;
>
> struct fs3270 {
> struct raw3270_view view;
> - struct pid *fs_pid; /* Pid of controlling program. */
> + struct task_ref *fs_tref; /* Pid of controlling program. */

```

```

> int read_command; /* ccw command to use for reads. */
> int write_command; /* ccw command to use for writes. */
> int attention; /* Got attention. */
> @@ -484,7 +484,7 @@ fs3270_close(struct inode *inode, struct
> fp = filp->private_data;
> filp->private_data = NULL;
> if (fp) {
> - put_pid(fp->fs_pid);
> + put_task_ref(fp->fs_pid);
> fp->fs_pid = NULL;
> raw3270_reset(&fp->view);
> raw3270_put_view(&fp->view);
> diff -puN drivers/usb/core/devio.c~rename-struct-pid drivers/usb/core/devio.c
> --- lxc/drivers/usb/core/devio.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/usb/core/devio.c 2007-04-10 16:18:30.000000000 -0700
> @@ -64,7 +64,7 @@ DEFINE_MUTEX(usbfs_mutex);
> struct async {
> struct list_head asynclist;
> struct dev_state *ps;
> - struct pid *pid;
> + struct task_ref *tref;
> uid_t uid, euid;
> unsigned int signr;
> unsigned int ifnum;
> @@ -224,7 +224,7 @@ static struct async *alloc_async(unsigned
>
> static void free_async(struct async *as)
> {
> - put_pid(as->pid);
> + put_task_ref(as->tref);
> kfree(as->urb->transfer_buffer);
> kfree(as->urb->setup_packet);
> usb_free_urb(as->urb);
> @@ -317,7 +317,7 @@ static void async_completed(struct urb *
> sinfo.si_errno = as->urb->status;
> sinfo.si_code = SI_ASYNCIO;
> sinfo.si_addr = as->userurb;
> - kill_pid_info_as_uid(as->signr, &sinfo, as->pid, as->uid,
> + kill_pid_info_as_uid(as->signr, &sinfo, as->tref, as->uid,
> as->euid, as->secid);
> }
> snoop(&urb->dev->dev, "urb complete\n");
> @@ -580,7 +580,7 @@ static int usbdev_open(struct inode *ino
> INIT_LIST_HEAD(&ps->async_completed);
> init_waitqueue_head(&ps->wait);
> ps->discsignr = 0;
> - ps->disc_pid = get_pid(task_pid(current));
> + ps->disc_tref = get_pid(task_pid(current));

```

```

> ps->disc_uid = current->uid;
> ps->disc_euid = current->euid;
> ps->disccontext = NULL;
> @@ -618,7 +618,7 @@ static int usbdev_release(struct inode *
> usb_autosuspend_device(dev);
> usb_unlock_device(dev);
> usb_put_dev(dev);
> - put_pid(ps->disc_pid);
> + put_task_ref(ps->disc_tref);
> kfree(ps);
> return 0;
> }
> @@ -1074,7 +1074,7 @@ static int proc_do_submiturb(struct dev_
> as->userbuffer = NULL;
> as->signr = uurb->signr;
> as->ifnum = ifnum;
> - as->pid = get_pid(task_pid(current));
> + as->tref = get_pid(task_pid(current));
> as->uid = current->uid;
> as->euid = current->euid;
> security_task_getsecid(current, &as->secid);
> diff -puN drivers/usb/core/inode.c~rename-struct-pid drivers/usb/core/inode.c
> --- lxc/drivers/usb/core/inode.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/usb/core/inode.c 2007-04-10 16:18:30.000000000 -0700
> @@ -698,7 +698,7 @@ static void usbfs_remove_device(struct u
> sinfo.si_errno = EPIPE;
> sinfo.si_code = SI_ASYNCIO;
> sinfo.si_addr = ds->disccontext;
> - kill_pid_info_as_uid(ds->discsignr, &sinfo, ds->disc_pid, ds->disc_uid, ds->disc_euid,
ds->secid);
> + kill_pid_info_as_uid(ds->discsignr, &sinfo, ds->disc_tref, ds->disc_uid, ds->disc_euid,
ds->secid);
> }
> }
> }
> diff -puN drivers/usb/core/usb.h~rename-struct-pid drivers/usb/core/usb.h
> --- lxc/drivers/usb/core/usb.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/drivers/usb/core/usb.h 2007-04-10 16:18:30.000000000 -0700
> @@ -135,7 +135,7 @@ struct dev_state {
> struct list_head async_completed;
> wait_queue_head_t wait; /* wake up if a request completed */
> unsigned int discsignr;
> - struct pid *disc_pid;
> + struct task_ref *disc_tref;
> uid_t disc_uid, disc_euid;
> void __user *disccontext;
> unsigned long ifclaimed;
> diff -puN files~rename-struct-pid files

```



```

> --- lxc/files~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/files 2007-04-10 16:18:30.000000000 -0700
> @@ -1,38 +1,4 @@
> -drivers/net/tun.c tun_chr_fasync 716 ret = __f_setown(file, task_pid(current), PIDTYPE_PID,
0);
> -fs/dnotify.c fcntl_dirnotify 95 error = __f_setown(filp, task_pid(current), PIDTYPE_PID, 0);
> -fs/fcntl.c f_setown 286 type = PIDTYPE_PID;
> -fs/fcntl.c f_setown 288 type = PIDTYPE_PGID;
> -fs/fcntl.c f_delown 301 f_modown(filp, NULL, PIDTYPE_PID, 0, 0, 1);
> -fs/fcntl.c f_getown 309 if (filp->f_owner.task_ref_type == PIDTYPE_PGID)
> -fs/ioprio.c sys_ioprio_set 105 do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> -fs/ioprio.c sys_ioprio_set 109 } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> -fs/ioprio.c sys_ioprio_get 192 do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> -fs/ioprio.c sys_ioprio_get 200 } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> -fs/locks.c fcntl_setlease 1517 error = __f_setown(filp, task_pid(current), PIDTYPE_PID, 0);
> -fs/proc/base.c proc_loginuid_write 815 if (current != pid_task(proc_pid(inode), PIDTYPE_PID))
> -fs/proc/base.c proc_pid_make_inode 1096 ei->pid = get_task_pid(task, PIDTYPE_PID);
> -fs/proc/base.c pid_getattr 1125 task = pid_task(proc_pid(inode), PIDTYPE_PID);
> -fs/proc/base.c pid_delete_dentry 1182 return
!proc_pid(dentry->d_inode)->tasks[PIDTYPE_PID].first;
> -fs/proc/base.c proc_base_instantiate 1919 ei->pid = get_task_pid(task, PIDTYPE_PID);
> -fs/proc/base.c next_tgid 2228 task = pid_task(pid, PIDTYPE_PID);
> -fs/proc/internal.h get_proc_task 69 return get_pid_task(proc_pid(inode), PIDTYPE_PID);
> -fs/proc/task_mmu.c m_start 396 priv->task = get_pid_task(priv->pid, PIDTYPE_PID);
> -fs/proc/task_nommu.c m_start 163 priv->task = get_pid_task(priv->pid, PIDTYPE_PID);
> -kernel/capability.c cap_set_pg 105 do_each_pid_task(pgrp, PIDTYPE_PGID, g) {
> -kernel/capability.c cap_set_pg 118 } while_each_pid_task(pgrp, PIDTYPE_PGID, g);
> -kernel/cpuset.c proc_cpuset_show 2553 tsk = get_pid_task(pid, PIDTYPE_PID);
> -kernel/futex.c futex_fd 2082 err = __f_setown(filp, task_pid(current), PIDTYPE_PID, 1);
> -kernel/pid.c alloc_pid 222 for (type = 0; type < PIDTYPE_MAX; ++type)
> -kernel/pid.c detach_pid 279 for (tmp = PIDTYPE_MAX; --tmp >= 0; )
> -kernel/signal.c __kill_pgrp_info 1234 do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> -kernel/signal.c __kill_pgrp_info 1238 } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> -kernel/signal.c kill_pid_info 1262 p = pid_task(pid, PIDTYPE_PID);
> -kernel/signal.c kill_pid_info_as_uid 1294 p = pid_task(pid, PIDTYPE_PID);
> -kernel/sys.c sys_setpriority 687 do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> -kernel/sys.c sys_setpriority 689 } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> -kernel/sys.c sys_getpriority 747 do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> -kernel/sys.c sys_getpriority 751 } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> -kernel/sys.c sys_setpgid 1477 find_task_by_pid_type(PIDTYPE_PGID, pgid);
> -kernel/sys.c sys_setpgid 1488 detach_pid(p, PIDTYPE_PGID);
> -kernel/sys.c sys_setpgid 1490 attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
> -kernel/sys.c sys_setsid 1574 if (session > 1 && find_task_by_pid_type(PIDTYPE_PGID,
session))
> +include/linux/sched.h <global> 1311 extern struct task_struct *find_task_by_tref_type(int type,
int pid);
> +include/linux/sched.h find_task_by_tref 1310 #define find_task_by_tref(nr)
find_task_by_tref_type(REFTYPE_PID, nr)

```

```

> +kernel/sys.c sys_setpgid 1477 find_task_by_tref_type(REFTYPE_PGID, pgid);
> +kernel/sys.c sys_setsid 1574 if (session > 1 && find_task_by_tref_type(REFTYPE_PGID,
session))
> diff -puN fs/autofs/autofs_i.h~rename-struct-pid fs/autofs/autofs_i.h
> --- lxc/fs/autofs/autofs_i.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/autofs/autofs_i.h 2007-04-10 16:18:30.000000000 -0700
> @@ -101,7 +101,7 @@ struct autofs_symlink {
> struct autofs_sb_info {
>     u32 magic;
>     struct file *pipe;
> - struct pid *oz_pgrp;
> + struct task_ref *oz_pgrp;
>     int catatonic;
>     struct super_block *sb;
>     unsigned long exp_timeout;
> diff -puN fs/autofs/inode.c~rename-struct-pid fs/autofs/inode.c
> --- lxc/fs/autofs/inode.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/autofs/inode.c 2007-04-10 16:18:30.000000000 -0700
> @@ -37,7 +37,7 @@ void autofs_kill_sb(struct super_block *
>     if (!sbi->catatonic)
>         autofs_catatonic_mode(sbi); /* Free wait queues, close pipe */
>
> - put_pid(sbi->oz_pgrp);
> + put_task_ref(sbi->oz_pgrp);
>
>     autofs_hash_nuke(sbi);
>     for (n = 0; n < AUTOFS_MAX_SYMLINKS; n++) {
> @@ -216,7 +216,7 @@ fail_fput:
>     printk("autofs: pipe file descriptor does not contain proper ops\n");
>     fput(pipe);
> fail_put_pid:
> - put_pid(sbi->oz_pgrp);
> + put_task_ref(sbi->oz_pgrp);
> fail_dput:
>     dput(root);
>     goto fail_free;
> diff -puN fs/autofs/root.c~rename-struct-pid fs/autofs/root.c
> --- lxc/fs/autofs/root.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/autofs/root.c 2007-04-10 16:18:30.000000000 -0700
> @@ -214,7 +214,7 @@ static struct dentry *autofs_root_lookup
>
>     oz_mode = autofs_oz_mode(sbi);
>     DPRINTK(("autofs_lookup: pid = %u, pgrp = %u, catatonic = %d, "
> -     "oz_mode = %d\n", pid_nr(task_pid(current)),
> +     "oz_mode = %d\n", tref_to_pid(task_pid(current)),
>     process_group(current), sbi->catatonic,
>     oz_mode));
>

```

```

> diff -puN fs/compat.c~rename-struct-pid fs/compat.c
> --- lxc/fs/compat.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/compat.c 2007-04-10 16:18:30.000000000 -0700
> @@ -738,7 +738,7 @@ static void *do_ncp_super_data_conv(void
>     n->gid = c_n->gid;
>     n->uid = c_n->uid;
>     memmove (n->mounted_vol, c_n->mounted_vol, (sizeof (c_n->mounted_vol) + 3 * sizeof
(unsigned int)));
> - n->wdog_pid = c_n->wdog_pid;
> + n->wdog_tref = c_n->wdog_tref;
>     n->mounted_uid = c_n->mounted_uid;
> } else if (version == 4) {
>     struct compat_ncp_mount_data_v4 *c_n = raw_data;
> @@ -751,7 +751,7 @@ static void *do_ncp_super_data_conv(void
>     n->retry_count = c_n->retry_count;
>     n->time_out = c_n->time_out;
>     n->ncp_fd = c_n->ncp_fd;
> - n->wdog_pid = c_n->wdog_pid;
> + n->wdog_tref = c_n->wdog_tref;
>     n->mounted_uid = c_n->mounted_uid;
>     n->flags = c_n->flags;
> } else if (version != 5) {
> diff -puN fs/dnotify.c~rename-struct-pid fs/dnotify.c
> --- lxc/fs/dnotify.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/dnotify.c 2007-04-10 16:18:30.000000000 -0700
> @@ -92,7 +92,7 @@ int fcntl_dirnotify(int fd, struct file
>     prev = &odn->dn_next;
> }
>
> - error = __f_setown(filp, task_pid(current), PIDTYPE_PID, 0);
> + error = __f_setown(filp, task_pid(current), REFTYPE_PID, 0);
> if (error)
>     goto out_free;
>
> diff -puN fs/exec.c~rename-struct-pid fs/exec.c
> --- lxc/fs/exec.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/exec.c 2007-04-10 16:18:30.000000000 -0700
> @@ -708,11 +708,11 @@ static int de_thread(struct task_struct
>     * Note: The old leader also uses this pid until release_task
>     *       is called. Odd but simple and correct.
>     */
> - detach_pid(tsk, PIDTYPE_PID);
> + detach_task_ref(tsk, REFTYPE_PID);
>     tsk->pid = leader->pid;
> - attach_pid(tsk, PIDTYPE_PID, find_pid(tsk->pid));
> - transfer_pid(leader, tsk, PIDTYPE_PGID);
> - transfer_pid(leader, tsk, PIDTYPE_SID);
> + attach_task_ref(tsk, REFTYPE_PID, find_task(tsk->pid));

```

```

> + transfer_task_ref(leader, tsk, REFTYPE_PGID);
> + transfer_task_ref(leader, tsk, REFTYPE_SID);
> list_replace_rcu(&leader->tasks, &tsk->tasks);
>
> tsk->group_leader = tsk;
> diff -puN fs/fcntl.c~rename-struct-pid fs/fcntl.c
> --- lxc/fs/fcntl.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/fcntl.c 2007-04-10 16:18:30.000000000 -0700
> @@ -249,21 +249,22 @@ static int setfl(int fd, struct file * f
> return error;
> }
>
> -static void f_modown(struct file *filp, struct pid *pid, enum pid_type type,
> - uid_t uid, uid_t euid, int force)
> +static void f_modown(struct file *filp, struct task_ref *tref,
> + enum task_ref_type type,
> + uid_t uid, uid_t euid, int force)
> {
> write_lock_irq(&filp->f_owner.lock);
> - if (force || !filp->f_owner.pid) {
> - put_pid(filp->f_owner.pid);
> - filp->f_owner.pid = get_pid(pid);
> - filp->f_owner.pid_type = type;
> + if (force || !filp->f_owner.tref) {
> + put_task_ref(filp->f_owner.tref);
> + filp->f_owner.tref = get_pid(tref);
> + filp->f_owner.task_ref_type = type;
> filp->f_owner.uid = uid;
> filp->f_owner.euid = euid;
> }
> write_unlock_irq(&filp->f_owner.lock);
> }
>
> -int __f_setown(struct file *filp, struct pid *pid, enum pid_type type,
> +int __f_setown(struct file *filp, struct task_ref *tref, enum task_ref_type type,
> int force)
> {
> int err;
> @@ -272,25 +273,25 @@ int __f_setown(struct file *filp, struct
> if (err)
> return err;
>
> - f_modown(filp, pid, type, current->uid, current->euid, force);
> + f_modown(filp, tref, type, current->uid, current->euid, force);
> return 0;
> }
> EXPORT_SYMBOL(__f_setown);
>

```

```

> int f_setown(struct file *filp, unsigned long arg, int force)
> {
> - enum pid_type type;
> - struct pid *pid;
> + enum task_ref_type type;
> + struct task_ref *tref;
> int who = arg;
> int result;
> - type = PIDTYPE_PID;
> + type = REFTYPE_PID;
> if (who < 0) {
> - type = PIDTYPE_PGID;
> + type = REFTYPE_PGID;
> who = -who;
> }
> rcu_read_lock();
> - pid = find_pid(who);
> - result = __f_setown(filp, pid, type, force);
> + tref = find_task(who);
> + result = __f_setown(filp, tref, type, force);
> rcu_read_unlock();
> return result;
> }
> @@ -298,15 +299,15 @@ EXPORT_SYMBOL(f_setown);
>
> void f_delown(struct file *filp)
> {
> - f_modown(filp, NULL, PIDTYPE_PID, 0, 0, 1);
> + f_modown(filp, NULL, REFTYPE_PID, 0, 0, 1);
> }
>
> pid_t f_getown(struct file *filp)
> {
> pid_t pid;
> read_lock(&filp->f_owner.lock);
> - pid = pid_nr(filp->f_owner.pid);
> - if (filp->f_owner.pid_type == PIDTYPE_PGID)
> + pid = tref_to_pid(filp->f_owner.tref);
> + if (filp->f_owner.task_ref_type == REFTYPE_PGID)
> pid = -pid;
> read_unlock(&filp->f_owner.lock);
> return pid;
> @@ -501,19 +502,19 @@ static void send_sigio_to_task(struct ta
> void send_sigio(struct fown_struct *fown, int fd, int band)
> {
> struct task_struct *p;
> - enum pid_type type;
> - struct pid *pid;

```

```

> + enum task_ref_type type;
> + struct task_ref *tref;
>
> read_lock(&fown->lock);
> - type = fown->pid_type;
> - pid = fown->pid;
> - if (!pid)
> + type = fown->task_ref_type;
> + tref = fown->tref;
> + if (!tref)
> goto out_unlock_fown;
>
> read_lock(&tasklist_lock);
> - do_each_pid_task(pid, type, p) {
> + do_each_referenced_task(tref, type, p) {
> send_sigurg_to_task(p, fown, fd, band);
> - } while_each_pid_task(pid, type, p);
> + } while_each_referenced_task(tref, type, p);
> read_unlock(&tasklist_lock);
> out_unlock_fown:
> read_unlock(&fown->lock);
> @@ -529,22 +530,22 @@ static void send_sigurg_to_task(struct t
> int send_sigurg(struct fown_struct *fown)
> {
> struct task_struct *p;
> - enum pid_type type;
> - struct pid *pid;
> + enum task_ref_type type;
> + struct task_ref *tref;
> int ret = 0;
>
> read_lock(&fown->lock);
> - type = fown->pid_type;
> - pid = fown->pid;
> - if (!pid)
> + type = fown->task_ref_type;
> + tref = fown->tref;
> + if (!tref)
> goto out_unlock_fown;
>
> ret = 1;
>
> read_lock(&tasklist_lock);
> - do_each_pid_task(pid, type, p) {
> + do_each_referenced_task(tref, type, p) {
> send_sigurg_to_task(p, fown);
> - } while_each_pid_task(pid, type, p);
> + } while_each_referenced_task(tref, type, p);

```

```

> read_unlock(&tasklist_lock);
> out_unlock_fown:
> read_unlock(&fown->lock);
> diff -puN fs/file_table.c~rename-struct-pid fs/file_table.c
> --- lxc/fs/file_table.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/file_table.c 2007-04-10 16:18:30.000000000 -0700
> @@ -175,7 +175,7 @@ void fastcall __fput(struct file *file)
>   fops_put(file->f_op);
>   if (file->f_mode & FMODE_WRITE)
>       put_write_access(inode);
> - put_pid(file->f_owner.pid);
> + put_task_ref(file->f_owner.tref);
>   file_kill(file);
>   file->f_path.dentry = NULL;
>   file->f_path.mnt = NULL;
> diff -puN fs/ioprio.c~rename-struct-pid fs/ioprio.c
> --- lxc/fs/ioprio.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/ioprio.c 2007-04-10 16:18:30.000000000 -0700
> @@ -60,7 +60,7 @@ asmlinkage long sys_ioprio_set(int which
>   int data = IOPRIO_PRIO_DATA(ioprio);
>   struct task_struct *p, *g;
>   struct user_struct *user;
> - struct pid *pgrp;
> + struct task_ref *pgrp;
>   int ret;
>
>   switch (class) {
> @@ -101,12 +101,12 @@ asmlinkage long sys_ioprio_set(int which
>   if (!who)
>       pgrp = task_pgrp(current);
>   else
> -   pgrp = find_pid(who);
> -   do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> +   pgrp = find_task(who);
> +   do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
>       ret = set_task_ioprio(p, ioprio);
>       if (ret)
>           break;
> -   } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> +   } while_each_referenced_task(pgrp, REFTYPE_PGID, p);
>       break;
>   case IOPRIO_WHO_USER:
>       if (!who)
> @@ -170,7 +170,7 @@ asmlinkage long sys_ioprio_get(int which
>   {
>       struct task_struct *g, *p;
>       struct user_struct *user;
> -   struct pid *pgrp;

```

```

> + struct task_ref *pgrp;
> int ret = -ESRCH;
> int tmpio;
>
> @@ -188,8 +188,8 @@ asmlinkage long sys_ioprio_get(int which
> if (!who)
> pgrp = task_pgrp(current);
> else
> - pgrp = find_pid(who);
> - do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> + pgrp = find_task(who);
> + do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
> tmpio = get_task_ioprio(p);
> if (tmpio < 0)
> continue;
> @@ -197,7 +197,7 @@ asmlinkage long sys_ioprio_get(int which
> ret = tmpio;
> else
> ret = ioprio_best(ret, tmpio);
> - } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, p);
> break;
> case IOPRIO_WHO_USER:
> if (!who)
> @@ -230,4 +230,3 @@ asmlinkage long sys_ioprio_get(int which
> read_unlock(&tasklist_lock);
> return ret;
> }
> -
> diff -puN fs/locks.c~rename-struct-pid fs/locks.c
> --- lxc/fs/locks.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/locks.c 2007-04-10 16:18:30.000000000 -0700
> @@ -1514,7 +1514,7 @@ int fcntl_setlease(unsigned int fd, stru
> goto out_unlock;
> }
>
> - error = __f_setown(filp, task_pid(current), PIDTYPE_PID, 0);
> + error = __f_setown(filp, task_pid(current), REFTYPE_PID, 0);
> out_unlock:
> unlock_kernel();
> return error;
> diff -puN fs/ncpfs/inode.c~rename-struct-pid fs/ncpfs/inode.c
> --- lxc/fs/ncpfs/inode.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/ncpfs/inode.c 2007-04-10 16:18:30.000000000 -0700
> @@ -332,7 +332,7 @@ static int ncp_parse_options(struct ncp_
> data->flags = 0;
> data->int_flags = 0;
> data->mounted_uid = 0;

```



```

> - data->wdog_pid = NULL;
> + data->wdog_tref = NULL;
> data->ncp_fd = ~0;
> data->time_out = 10;
> data->retry_count = 20;
> @@ -373,7 +373,7 @@ static int ncp_parse_options(struct ncp_
> data->flags = optint;
> break;
> case 'w':
> - data->wdog_pid = find_get_pid(optint);
> + data->wdog_tref = find_get_pid(optint);
> break;
> case 'n':
> data->ncp_fd = optint;
> @@ -394,8 +394,8 @@ static int ncp_parse_options(struct ncp_
> }
> return 0;
> err:
> - put_pid(data->wdog_pid);
> - data->wdog_pid = NULL;
> + put_task_ref(data->wdog_tref);
> + data->wdog_tref = NULL;
> return ret;
> }
>
> @@ -414,7 +414,7 @@ static int ncp_fill_super(struct super_b
> #endif
> struct ncp_entry_info finfo;
>
> - data.wdog_pid = NULL;
> + data.wdog_tref = NULL;
> server = kzalloc(sizeof(struct ncp_server), GFP_KERNEL);
> if (!server)
> return -ENOMEM;
> @@ -431,7 +431,7 @@ static int ncp_fill_super(struct super_b
> data.flags = md->flags;
> data.int_flags = NCP_IMOUNT_LOGGEDIN_POSSIBLE;
> data.mounted_uid = md->mounted_uid;
> - data.wdog_pid = find_get_pid(md->wdog_pid);
> + data.wdog_tref = find_get_pid(md->wdog_pid);
> data.ncp_fd = md->ncp_fd;
> data.time_out = md->time_out;
> data.retry_count = md->retry_count;
> @@ -451,7 +451,7 @@ static int ncp_fill_super(struct super_b
> data.flags = md->flags;
> data.int_flags = 0;
> data.mounted_uid = md->mounted_uid;
> - data.wdog_pid = find_get_pid(md->wdog_pid);

```

```

> + data.wdog_tref = find_get_pid(md->wdog_pid);
> data.ncp_fd = md->ncp_fd;
> data.time_out = md->time_out;
> data.retry_count = md->retry_count;
> @@ -695,7 +695,7 @@ out_fput:
> */
> fput(ncp_filp);
> out:
> - put_pid(data.wdog_pid);
> + put_task_ref(data.wdog_tref);
> sb->s_fs_info = NULL;
> kfree(server);
> return error;
> @@ -728,8 +728,8 @@ static void ncp_put_super(struct super_b
> if (server->info_filp)
> fput(server->info_filp);
> fput(server->ncp_filp);
> - kill_pid(server->m.wdog_pid, SIGTERM, 1);
> - put_pid(server->m.wdog_pid);
> + kill_pid(server->m.wdog_tref, SIGTERM, 1);
> + put_task_ref(server->m.wdog_tref);
>
> kfree(server->priv.data);
> kfree(server->auth.object_name);
> diff -puN fs/proc/array.c~rename-struct-pid fs/proc/array.c
> --- lxc/fs/proc/array.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/array.c 2007-04-10 16:18:30.000000000 -0700
> @@ -355,7 +355,7 @@ static int do_task_stat(struct task_stru
> struct signal_struct *sig = task->signal;
>
> if (sig->tty) {
> - tty_pgrp = pid_nr(sig->tty->pgrp);
> + tty_pgrp = tref_to_pid(sig->tty->pgrp);
> tty_nr = new_encode_dev(tty_devnum(sig->tty));
> }
>
> diff -puN fs/proc/base.c~rename-struct-pid fs/proc/base.c
> --- lxc/fs/proc/base.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/base.c 2007-04-10 16:18:30.000000000 -0700
> @@ -812,7 +812,7 @@ static ssize_t proc_loginuid_write(struc
> if (!capable(CAP_AUDIT_CONTROL))
> return -EPERM;
>
> - if (current != pid_task(proc_pid(inode), PIDTYPE_PID))
> + if (current != pid_task(proc_task_ref(inode), REFTYPE_PID))
> return -EPERM;
>
> if (count >= PAGE_SIZE)

```

```

> @@ -1093,8 +1093,8 @@ static struct inode *proc_pid_make_inode
> /*
>  * grab the reference to task.
>  */
> - ei->pid = get_task_pid(task, PIDTYPE_PID);
> - if (!ei->pid)
> + ei->tref = get_task_ref(task, REFTYPE_PID);
> + if (!ei->tref)
>     goto out_unlock;
>
>     inode->i_uid = 0;
> @@ -1122,7 +1122,7 @@ static int pid_getattr(struct vfsmount *
>     rcu_read_lock();
>     stat->uid = 0;
>     stat->gid = 0;
> - task = pid_task(proc_pid(inode), PIDTYPE_PID);
> + task = pid_task(proc_task_ref(inode), REFTYPE_PID);
>     if (task) {
>         if ((inode->i_mode == (S_IFDIR|S_IRUGO|S_IXUGO)) ||
>             task_dumpable(task)) {
> @@ -1179,7 +1179,7 @@ static int pid_delete_dentry(struct dent
>     * If so, then don't put the dentry on the lru list,
>     * kill it immediately.
>     */
> - return !proc_pid(dentry->d_inode)->tasks[PIDTYPE_PID].first;
> + return !proc_task_ref(dentry->d_inode)->tasks[REFTYPE_PID].first;
> }
>
> static struct dentry_operations pid_dentry_operations =
> @@ -1537,7 +1537,7 @@ static int proc_fd_permission(struct ino
>     rv = generic_permission(inode, mask, NULL);
>     if (rv == 0)
>         return 0;
> - if (task_pid(current) == proc_pid(inode))
> + if (task_pid(current) == proc_task_ref(inode))
>     rv = 0;
>     return rv;
> }
> @@ -1603,7 +1603,7 @@ static const struct inode_operations pro
> };
>
>
> -static struct dentry *proc_pident_instantiate(struct inode *dir,
> +static struct dentry *proc_task_refent_instantiate(struct inode *dir,
>     struct dentry *dentry, struct task_struct *task, const void *ptr)
> {
>     const struct pid_entry *p = ptr;
> @@ -1633,7 +1633,7 @@ out:

```

```

> return error;
> }
>
> -static struct dentry *proc_pident_lookup(struct inode *dir,
> +static struct dentry *proc_task_refent_lookup(struct inode *dir,
>     struct dentry *dentry,
>     const struct pid_entry *ents,
>     unsigned int nents)
> @@ -1663,21 +1663,21 @@ static struct dentry *proc_pident_lookup
> if (p > last)
>     goto out;
>
> - error = proc_pident_instantiate(dir, dentry, task, p);
> + error = proc_task_refent_instantiate(dir, dentry, task, p);
> out:
> put_task_struct(task);
> out_no_task:
> return error;
> }
>
> -static int proc_pident_fill_cache(struct file *filp, void *dirent,
> +static int proc_task_refent_fill_cache(struct file *filp, void *dirent,
>     filldir_t filldir, struct task_struct *task, const struct pid_entry *p)
> {
>     return proc_fill_cache(filp, dirent, filldir, p->name, p->len,
> -     proc_pident_instantiate, task, p);
> +     proc_task_refent_instantiate, task, p);
> }
>
> -static int proc_pident_readdir(struct file *filp,
> +static int proc_task_refent_readdir(struct file *filp,
>     void *dirent, filldir_t filldir,
>     const struct pid_entry *ents, unsigned int nents)
> {
> @@ -1721,7 +1721,7 @@ static int proc_pident_readdir(struct fi
>     p = ents + i;
>     last = &ents[nents - 1];
>     while (p <= last) {
> -     if (proc_pident_fill_cache(filp, dirent, filldir, task, p) < 0)
> +     if (proc_task_refent_fill_cache(filp, dirent, filldir, task, p) < 0)
>         goto out;
>         filp->f_pos++;
>         p++;
> @@ -1813,7 +1813,7 @@ static const struct pid_entry attr_dir_s
> static int proc_attr_dir_readdir(struct file * filp,
>     void * dirent, filldir_t filldir)
> {
> - return proc_pident_readdir(filp,dirent,filldir,

```

```

> + return proc_task_refent_readdir(filp,dirent,filldir,
>     attr_dir_stuff,ARRAY_SIZE(attr_dir_stuff));
> }
>
> @@ -1825,7 +1825,7 @@ static const struct file_operations proc
> static struct dentry *proc_attr_dir_lookup(struct inode *dir,
>     struct dentry *dentry, struct nameidata *nd)
> {
> - return proc_pident_lookup(dir, dentry,
> + return proc_task_refent_lookup(dir, dentry,
>     attr_dir_stuff, ARRAY_SIZE(attr_dir_stuff));
> }
>
> @@ -1916,8 +1916,8 @@ static struct dentry *proc_base_instanti
> /*
>  * grab the reference to the task.
>  */
> - ei->pid = get_task_pid(task, PIDTYPE_PID);
> - if (!ei->pid)
> + ei->tref = get_task_ref(task, REFTYPE_PID);
> + if (!ei->tref)
>     goto out_iput;
>
> inode->i_uid = 0;
> @@ -2065,7 +2065,7 @@ static const struct pid_entry tgid_base_
> static int proc_tgid_base_readdir(struct file * filp,
>     void * dirent, filldir_t filldir)
> {
> - return proc_pident_readdir(filp,dirent,filldir,
> + return proc_task_refent_readdir(filp,dirent,filldir,
>     tgid_base_stuff,ARRAY_SIZE(tgid_base_stuff));
> }
>
> @@ -2075,7 +2075,7 @@ static const struct file_operations proc
> };
>
> static struct dentry *proc_tgid_base_lookup(struct inode *dir, struct dentry *dentry, struct
nameidata *nd){
> - return proc_pident_lookup(dir, dentry,
> + return proc_task_refent_lookup(dir, dentry,
>     tgid_base_stuff, ARRAY_SIZE(tgid_base_stuff));
> }
>
> @@ -2217,15 +2217,15 @@ out:
> static struct task_struct *next_tgid(unsigned int tgid)
> {
>     struct task_struct *task;
> - struct pid *pid;

```

```

> + struct task_ref *tref;
>
> rcu_read_lock();
> retry:
> task = NULL;
> - pid = find_ge_pid(tgid);
> - if (pid) {
> - tgid = pid->nr + 1;
> - task = pid_task(pid, PIDTYPE_PID);
> + tref = find_ge_pid(tgid);
> + if (tref) {
> + tgid = tref->pid + 1;
> + task = pid_task(tref, REFTYPE_PID);
> /* What we to know is if the pid we have find is the
> * pid of a thread_group_leader. Testing for task
> * being a thread_group_leader is the obvious thing
> @@ -2345,12 +2345,12 @@ static const struct pid_entry tid_base_s
> static int proc_tid_base_readdir(struct file * filp,
> void * dirent, filldir_t filldir)
> {
> - return proc_pident_readdir(filp,dirent,filldir,
> + return proc_task_refent_readdir(filp,dirent,filldir,
> tid_base_stuff,ARRAY_SIZE(tid_base_stuff));
> }
>
> static struct dentry *proc_tid_base_lookup(struct inode *dir, struct dentry *dentry, struct
nameidata *nd){
> - return proc_pident_lookup(dir, dentry,
> + return proc_task_refent_lookup(dir, dentry,
> tid_base_stuff, ARRAY_SIZE(tid_base_stuff));
> }
>
> diff -puN fs/proc/inode.c~rename-struct-pid fs/proc/inode.c
> --- lxc/fs/proc/inode.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/inode.c 2007-04-10 16:18:30.000000000 -0700
> @@ -62,7 +62,7 @@ static void proc_delete_inode(struct ino
> truncate_inode_pages(&inode->i_data, 0);
>
> /* Stop tracking associated processes */
> - put_pid(PROC_I(inode)->pid);
> + put_task_ref(PROC_I(inode)->tref);
>
> /* Let go of any associated proc directory entry */
> de = PROC_I(inode)->pde;
> @@ -91,7 +91,7 @@ static struct inode *proc_alloc_inode(st
> ei = (struct proc_inode *)kmem_cache_alloc(proc_inode_cache, GFP_KERNEL);
> if (!ei)
> return NULL;

```

```

> - ei->pid = NULL;
> + ei->tref= NULL;
> ei->fd = 0;
> ei->op.proc_get_link = NULL;
> ei->pde = NULL;
> diff -puN fs/proc/internal.h~rename-struct-pid fs/proc/internal.h
> --- lxc/fs/proc/internal.h~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/internal.h 2007-04-10 16:18:30.000000000 -0700
> @@ -59,14 +59,14 @@ void free_proc_entry(struct proc_dir_ent
>
> int proc_init_inodecache(void);
>
> -static inline struct pid *proc_pid(struct inode *inode)
> +static inline struct task_ref *proc_task_ref(struct inode *inode)
> {
> - return PROC_I(inode)->pid;
> + return PROC_I(inode)->tref;
> }
>
> static inline struct task_struct *get_proc_task(struct inode *inode)
> {
> - return get_pid_task(proc_pid(inode), PIDTYPE_PID);
> + return get_pid_task(proc_task_ref(inode), REFTYPE_PID);
> }
>
> static inline int proc_fd(struct inode *inode)
> diff -puN fs/proc/root.c~rename-struct-pid fs/proc/root.c
> --- lxc/fs/proc/root.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/root.c 2007-04-10 16:18:30.000000000 -0700
> @@ -34,8 +34,8 @@ static int proc_get_sb(struct file_syste
> */
> struct proc_inode *ei;
> ei = PROC_I(proc_mnt->mnt_sb->s_root->d_inode);
> - if (!ei->pid)
> - ei->pid = find_get_pid(1);
> + if (!ei->tref)
> + ei->tref = find_get_pid(1);
> }
> return get_sb_single(fs_type, flags, data, proc_fill_super, mnt);
> }
> diff -puN fs/proc/task_mmu.c~rename-struct-pid fs/proc/task_mmu.c
> --- lxc/fs/proc/task_mmu.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/task_mmu.c 2007-04-10 16:18:30.000000000 -0700
> @@ -393,7 +393,7 @@ static void *m_start(struct seq_file *m,
> if (last_addr == -1UL)
> return NULL;
>
>
> - priv->task = get_pid_task(priv->pid, PIDTYPE_PID);

```

```

> + priv->task = get_pid_task(priv->tref, REFTYPE_PID);
> if (!priv->task)
> return NULL;
>
> @@ -489,7 +489,7 @@ static int do_maps_open(struct inode *in
> int ret = -ENOMEM;
> priv = kzalloc(sizeof(*priv), GFP_KERNEL);
> if (priv) {
> - priv->pid = proc_pid(inode);
> + priv->tref = proc_task_ref(inode);
> ret = seq_open(file, ops);
> if (!ret) {
> struct seq_file *m = file->private_data;
> diff -puN fs/proc/task_nommu.c~rename-struct-pid fs/proc/task_nommu.c
> --- lxc/fs/proc/task_nommu.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/proc/task_nommu.c 2007-04-10 16:18:30.000000000 -0700
> @@ -160,7 +160,7 @@ static void *m_start(struct seq_file *m,
> loff_t n = *pos;
>
> /* pin the task and mm whilst we play with them */
> - priv->task = get_pid_task(priv->pid, PIDTYPE_PID);
> + priv->task = get_pid_task(priv->pid, REFTYPE_PID);
> if (!priv->task)
> return NULL;
>
> @@ -214,7 +214,7 @@ static int maps_open(struct inode *inode
>
> priv = kzalloc(sizeof(*priv), GFP_KERNEL);
> if (priv) {
> - priv->pid = proc_pid(inode);
> + priv->pid = proc_task_ref(inode);
> ret = seq_open(file, &proc_pid_maps_ops);
> if (!ret) {
> struct seq_file *m = file->private_data;
> @@ -232,4 +232,3 @@ const struct file_operations proc_maps_o
> .llseek = seq_llseek,
> .release = seq_release_private,
> };
> -
> diff -puN fs/smbfs/inode.c~rename-struct-pid fs/smbfs/inode.c
> --- lxc/fs/smbfs/inode.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/smbfs/inode.c 2007-04-10 16:18:30.000000000 -0700
> @@ -481,14 +481,14 @@ smb_put_super(struct super_block *sb)
>
> smb_close_socket(server);
>
> - if (server->conn_pid)
> - kill_pid(server->conn_pid, SIGTERM, 1);

```



```

> + if (server->conn_tref)
> + kill_pid(server->conn_tref, SIGTERM, 1);
>
> kfree(server->ops);
> smb_unload_nls(server);
> sb->s_fs_info = NULL;
> smb_unlock_server(server);
> - put_pid(server->conn_pid);
> + put_task_ref(server->conn_tref);
> kfree(server);
> }
>
> @@ -538,7 +538,7 @@ static int smb_fill_super(struct super_b
> INIT_LIST_HEAD(&server->xmitq);
> INIT_LIST_HEAD(&server->recvq);
> server->conn_error = 0;
> - server->conn_pid = NULL;
> + server->conn_tref = NULL;
> server->state = CONN_INVALID; /* no connection yet */
> server->generation = 0;
>
> diff -puN fs/smbfs/proc.c~rename-struct-pid fs/smbfs/proc.c
> --- lxc/fs/smbfs/proc.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/smbfs/proc.c 2007-04-10 16:18:30.000000000 -0700
> @@ -877,7 +877,7 @@ smb_newconn(struct smb_sb_info *server,
> goto out_putf;
>
> server->sock_file = filp;
> - server->conn_pid = get_pid(task_pid(current));
> + server->conn_tref = get_pid(task_pid(current));
> server->opt = *opt;
> server->generation += 1;
> server->state = CONN_VALID;
> @@ -972,7 +972,7 @@ smb_newconn(struct smb_sb_info *server,
>
> VERBOSE("protocol=%d, max_xmit=%d, pid=%d capabilities=0x%x\n",
> server->opt.protocol, server->opt.max_xmit,
> - pid_nr(server->conn_pid), server->opt.capabilities);
> + tref_to_pid(server->conn_tgid), server->opt.capabilities);
>
> /* FIXME: this really should be done by smbmount. */
> if (server->opt.max_xmit > SMB_MAX_PACKET_SIZE) {
> diff -puN fs/smbfs/smbiod.c~rename-struct-pid fs/smbfs/smbiod.c
> --- lxc/fs/smbfs/smbiod.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/fs/smbfs/smbiod.c 2007-04-10 16:18:30.000000000 -0700
> @@ -151,7 +151,7 @@ int smbiod_retry(struct smb_sb_info *ser
> {
> struct list_head *head;

```

```

> struct smb_request *req;
> - struct pid *pid = get_pid(server->conn_pid);
> + struct task_ref *tref = get_pid(server->conn_tref);
> int result = 0;
>
> VERBOSE("state: %d\n", server->state);
> @@ -206,7 +206,7 @@ int smbiod_retry(struct smb_sb_info *ser
>
> smb_close_socket(server);
>
> - if (pid == 0) {
> + if (tref == 0) {
>   /* FIXME: this is fatal, umount? */
>   printk(KERN_ERR "smb_retry: no connection process\n");
>   server->state = CONN_RETRIED;
> @@ -221,18 +221,18 @@ int smbiod_retry(struct smb_sb_info *ser
>   /*
>    * Note: use the "priv" flag, as a user process may need to reconnect.
>    */
>   - result = kill_pid(pid, SIGUSR1, 1);
>   + result = kill_pid(tref, SIGUSR1, 1);
>   if (result) {
>     /* FIXME: this is most likely fatal, umount? */
>     printk(KERN_ERR "smb_retry: signal failed [%d]\n", result);
>     goto out;
>   }
>   - VERBOSE("signalled pid %d\n", pid);
>   + VERBOSE("signalled task reference %p\n", tref);
>
>   /* FIXME: The retried requests should perhaps get a "time boost". */
>
> out:
> - put_pid(pid);
> + put_task_ref(tref);
> return result;
> }
>
> diff -puN init/main.c~rename-struct-pid init/main.c
> --- lxc/init/main.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/init/main.c 2007-04-10 16:18:30.000000000 -0700
> @@ -801,7 +801,7 @@ static int __init kernel_init(void * unu
>   init_pid_ns.child_reaper = current;
>
>   __set_special_pids(1, 1);
>   - cad_pid = task_pid(current);
>   + cad_tref = task_pid(current);
>
>   smp_prepare_cpus(max_cpus);

```

```

>
> diff -puN ipc/mqueue.c~rename-struct-pid ipc/mqueue.c
> --- lxc/ipc/mqueue.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/ipc/mqueue.c 2007-04-10 16:18:30.000000000 -0700
> @@ -73,7 +73,7 @@ struct mqueue_inode_info {
> struct mq_attr attr;
>
> struct sigevent notify;
> - struct pid* notify_owner;
> + struct task_ref* notify_owner;
> struct user_struct *user; /* user who created, for accounting */
> struct sock *notify_sock;
> struct sk_buff *notify_cookie;
> @@ -338,7 +338,7 @@ static ssize_t mqueue_read_file(struct f
> (info->notify_owner &&
> info->notify.sigev_notify == SIGEV_SIGNAL) ?
> info->notify.sigev_signo : 0,
> - pid_nr(info->notify_owner));
> + tref_to_pid(info->notify_owner));
> spin_unlock(&info->lock);
> buffer[sizeof(buffer)-1] = '\0';
> slen = strlen(buffer)+1;
> @@ -528,7 +528,7 @@ static void __do_notify(struct mqueue_in
> break;
> }
> /* after notification unregisters process */
> - put_pid(info->notify_owner);
> + put_task_ref(info->notify_owner);
> info->notify_owner = NULL;
> }
> wake_up(&info->wait_q);
> @@ -572,7 +572,7 @@ static void remove_notification(struct m
> set_cookie(info->notify_cookie, NOTIFY_REMOVED);
> netlink_sendskb(info->notify_sock, info->notify_cookie, 0);
> }
> - put_pid(info->notify_owner);
> + put_task_ref(info->notify_owner);
> info->notify_owner = NULL;
> }
>
> diff -puN kernel/capability.c~rename-struct-pid kernel/capability.c
> --- lxc/kernel/capability.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/capability.c 2007-04-10 16:18:30.000000000 -0700
> @@ -99,10 +99,10 @@ static inline int cap_set_pg(int pgrp_nr
> struct task_struct *g, *target;
> int ret = -EPERM;
> int found = 0;
> - struct pid *pgrp;

```

```

> + struct task_ref *pgrp;
>
> - pgrp = find_pid(pgrp_nr);
> - do_each_pid_task(pgrp, PIDTYPE_PGID, g) {
> + pgrp = find_task(pgrp_nr);
> + do_each_referenced_task(pgrp, REFTYPE_PGID, g) {
>     target = g;
>     while_each_thread(g, target) {
>         if (!security_capset_check(target, effective,
> @@ -115,7 +115,7 @@ static inline int cap_set_pg(int pgrp_nr
>     }
>     found = 1;
> }
> - } while_each_pid_task(pgrp, PIDTYPE_PGID, g);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, g);
>
> if (!found)
>     ret = 0;
> diff -puN kernel/cpuset.c~rename-struct-pid kernel/cpuset.c
> --- lxc/kernel/cpuset.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/cpuset.c 2007-04-10 16:18:30.000000000 -0700
> @@ -2538,7 +2538,7 @@ void __cpuset_memory_pressure_bump(void)
> */
> static int proc_cpuset_show(struct seq_file *m, void *v)
> {
> - struct pid *pid;
> + struct task_ref *tref;
>     struct task_struct *tsk;
>     char *buf;
>     int retval;
> @@ -2549,8 +2549,8 @@ static int proc_cpuset_show(struct seq_f
>     goto out;
>
>     retval = -ESRCH;
> - pid = m->private;
> - tsk = get_pid_task(pid, PIDTYPE_PID);
> + tref = m->private;
> + tsk = get_pid_task(tref, REFTYPE_PID);
>     if (!tsk)
>         goto out_free;
>
> @@ -2573,8 +2573,8 @@ out:
>
> static int cpuset_open(struct inode *inode, struct file *file)
> {
> - struct pid *pid = PROC_I(inode)->pid;
> - return single_open(file, proc_cpuset_show, pid);
> + struct task_ref *tref = PROC_I(inode)->tref;

```

```

> + return single_open(file, proc_cpuset_show, tref);
> }
>
> const struct file_operations proc_cpuset_operations = {
> diff -puN kernel/exit.c~rename-struct-pid kernel/exit.c
> --- lxc/kernel/exit.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/exit.c 2007-04-10 16:18:30.000000000 -0700
> @@ -57,10 +57,10 @@ static void exit_mm(struct task_struct *
> static void __unhash_process(struct task_struct *p)
> {
>     nr_threads--;
>     detach_pid(p, PIDTYPE_PID);
>     detach_task_ref(p, REFTYPE_PID);
>     if (thread_group_leader(p)) {
>         detach_pid(p, PIDTYPE_PGID);
>         detach_pid(p, PIDTYPE_SID);
>         detach_task_ref(p, REFTYPE_PGID);
>         detach_task_ref(p, REFTYPE_SID);
>
>         list_del_rcu(&p->tasks);
>         __get_cpu_var(process_counts)--;
> @@ -212,14 +212,14 @@ repeat:
> *
> * The caller must hold rcu lock or the tasklist lock.
> */
> -struct pid *session_of_pgrp(struct pid *pgrp)
> +struct task_ref *session_of_pgrp(struct task_ref *pgrp)
> {
>     struct task_struct *p;
>     - struct pid *sid = NULL;
>     + struct task_ref *sid = NULL;
>
>     - p = pid_task(pgrp, PIDTYPE_PGID);
>     + p = pid_task(pgrp, REFTYPE_PGID);
>     if (p == NULL)
>     - p = pid_task(pgrp, PIDTYPE_PID);
>     + p = pid_task(pgrp, REFTYPE_PID);
>     if (p != NULL)
>         sid = task_session(p);
>
> @@ -234,12 +234,12 @@ struct pid *session_of_pgrp(struct pid *
> *
> * "I ask you, have you ever known what it is to be an orphan?"
> */
> -static int will_become_orphaned_pgrp(struct pid *pgrp, struct task_struct *ignored_task)
> +static int will_become_orphaned_pgrp(struct task_ref *pgrp, struct task_struct *ignored_task)
> {
>     struct task_struct *p;

```

```

> int ret = 1;
>
> - do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> + do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
>   if (p == ignored_task
>       || p->exit_state
>       || is_init(p->parent))
> @@ -249,7 +249,7 @@ static int will_become_orphaned_pgrp(str
>   ret = 0;
>   break;
> }
> - } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, p);
> return ret; /* (sighing) "Often!" */
> }
>
> @@ -264,17 +264,17 @@ int is_current_pgrp_orphaned(void)
> return retval;
> }
>
> -static int has_stopped_jobs(struct pid *pgrp)
> +static int has_stopped_jobs(struct task_ref *pgrp)
> {
>   int retval = 0;
>   struct task_struct *p;
>
> - do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> + do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
>   if (p->state != TASK_STOPPED)
>     continue;
>   retval = 1;
>   break;
> - } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, p);
> return retval;
> }
>
> @@ -320,14 +320,14 @@ void __set_special_pids(pid_t session, p
> struct task_struct *curr = current->group_leader;
>
> if (process_session(curr) != session) {
> - detach_pid(curr, PIDTYPE_SID);
> + detach_task_ref(curr, REFTYPE_SID);
>   set_signal_session(curr->signal, session);
> - attach_pid(curr, PIDTYPE_SID, find_pid(session));
> + attach_task_ref(curr, REFTYPE_SID, find_task(session));
> }
> if (process_group(curr) != pgrp) {

```

```

> - detach_pid(curr, PIDTYPE_PGID);
> + detach_task_ref(curr, REFTYPE_PGID);
> curr->signal->pgrp = pgrp;
> - attach_pid(curr, PIDTYPE_PGID, find_pid(pgrp));
> + attach_task_ref(curr, REFTYPE_PGID, find_task(pgrp));
> }
> }
>
> @@ -651,7 +651,7 @@ reparent_thread(struct task_struct *p, s
> */
> if ((task_pgrp(p) != task_pgrp(father)) &&
>     (task_session(p) == task_session(father))) {
> - struct pid *pgrp = task_pgrp(p);
> + struct task_ref *pgrp = task_pgrp(p);
>
> if (will_become_orphaned_pgrp(pgrp, NULL) &&
>     has_stopped_jobs(pgrp)) {
> @@ -697,7 +697,7 @@ static void exit_notify(struct task_stru
> {
> int state;
> struct task_struct *t;
> - struct pid *pgrp;
> + struct task_ref *pgrp;
> int noreap;
> void *cookie;
>
> diff -puN kernel/fork.c~rename-struct-pid kernel/fork.c
> --- lxc/kernel/fork.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/fork.c 2007-04-10 16:18:30.000000000 -0700
> @@ -956,7 +956,7 @@ static struct task_struct *copy_process(
> unsigned long stack_size,
> int __user *parent_tidptr,
> int __user *child_tidptr,
> - struct pid *pid)
> + struct task_ref *tref)
> {
> int retval;
> struct task_struct *p = NULL;
> @@ -1023,7 +1023,7 @@ static struct task_struct *copy_process(
> p->did_exec = 0;
> delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
> copy_flags(clone_flags, p);
> - p->pid = pid_nr(pid);
> + p->pid = tref_to_pid(tref);
>
> INIT_LIST_HEAD(&p->children);
> INIT_LIST_HEAD(&p->sibling);
> @@ -1245,13 +1245,13 @@ static struct task_struct *copy_process(

```

```

> p->signal->tty = current->signal->tty;
> p->signal->pgrp = process_group(current);
> set_signal_session(p->signal, process_session(current));
> - attach_pid(p, PIDTYPE_PGID, task_pgrp(current));
> - attach_pid(p, PIDTYPE_SID, task_session(current));
> + attach_task_ref(p, REFTYPE_PGID, task_pgrp(current));
> + attach_task_ref(p, REFTYPE_SID, task_session(current));
>
> list_add_tail_rcu(&p->tasks, &init_task.tasks);
> __get_cpu_var(process_counts)++;
> }
> - attach_pid(p, PIDTYPE_PID, pid);
> + attach_task_ref(p, REFTYPE_PID, tref);
> nr_threads++;
> }
>
> @@ -1323,7 +1323,7 @@ struct task_struct * __cpuinit fork_idle
> struct pt_regs regs;
>
> task = copy_process(CLONE_VM, 0, idle_regs(&regs), 0, NULL, NULL,
> - &init_struct_pid);
> + &init_task_ref);
> if (!IS_ERR(task))
> init_idle(task, cpu);
>
> @@ -1344,14 +1344,15 @@ long do_fork(unsigned long clone_flags,
> int __user *child_tidptr)
> {
> struct task_struct *p;
> - struct pid *pid = alloc_pid();
> + struct task_ref *tref = alloc_task_ref();
> long nr;
>
> - if (!pid)
> + if (!tref)
> return -EAGAIN;
> - nr = pid->nr;
> + nr = tref->pid;
>
> - p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
> + p = copy_process(clone_flags, stack_start, regs, stack_size,
> + parent_tidptr, child_tidptr, tref);
> /*
> * Do this prior waking up the new thread - the thread pointer
> * might get invalid after that point, if the thread exits quickly.
> @@ -1393,7 +1394,7 @@ long do_fork(unsigned long clone_flags,
> tracehook_report_vfork_done(p, nr);
> }

```



```

> } else {
> - free_pid(pid);
> + free_task_ref(tref);
> nr = PTR_ERR(p);
> }
> return nr;
> diff -puN kernel/futex.c~rename-struct-pid kernel/futex.c
> --- lxc/kernel/futex.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/futex.c 2007-04-10 16:18:30.000000000 -0700
> @@ -2079,7 +2079,7 @@ static int futex_fd(u32 __user *uaddr, i
> filp->f_mapping = filp->f_path.dentry->d_inode->i_mapping;
>
> if (signal) {
> - err = __f_setown(filp, task_pid(current), PIDTYPE_PID, 1);
> + err = __f_setown(filp, task_pid(current), REFTYPE_PID, 1);
> if (err < 0) {
> goto error;
> }
> diff -puN kernel/pid.c~rename-struct-pid kernel/pid.c
> --- lxc/kernel/pid.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/pid.c 2007-04-10 16:18:30.000000000 -0700
> @@ -32,8 +32,8 @@
> #define pid_hashfn(nr) hash_long((unsigned long)nr, pidhash_shift)
> static struct hlist_head *pid_hash;
> static int pidhash_shift;
> -static struct kmem_cache *pid_cachep;
> -struct pid init_struct_pid = INIT_STRUCT_PID;
> +static struct kmem_cache *task_ref_cachep;
> +struct task_ref init_task_ref = INIT_TASK_REF;
>
> int pid_max = PID_MAX_DEFAULT;
>
> @@ -174,130 +174,131 @@ static int next_pidmap(struct pid_namesp
> return -1;
> }
>
> -fastcall void put_pid(struct pid *pid)
> +fastcall void put_task_ref(struct task_ref *tref)
> {
> - if (!pid)
> + if (!tref)
> return;
> - if ((atomic_read(&pid->count) == 1) ||
> - atomic_dec_and_test(&pid->count))
> - kmem_cache_free(pid_cachep, pid);
> + if ((atomic_read(&tref->count) == 1) ||
> + atomic_dec_and_test(&tref->count))
> + kmem_cache_free(task_ref_cachep, tref);

```

```

> }
> -EXPORT_SYMBOL_GPL(put_pid);
> +EXPORT_SYMBOL_GPL(put_task_ref);
>
> -static void delayed_put_pid(struct rcu_head *rhp)
> +static void delayed_put_task_ref(struct rcu_head *rhp)
> {
> - struct pid *pid = container_of(rhp, struct pid, rcu);
> - put_pid(pid);
> + struct task_ref *tref = container_of(rhp, struct task_ref, rcu);
> + put_task_ref(tref);
> }
>
> -fastcall void free_pid(struct pid *pid)
> +fastcall void free_task_ref(struct task_ref *tref)
> {
> /* We can be called with write_lock_irq(&tasklist_lock) held */
> unsigned long flags;
>
> spin_lock_irqsave(&pidmap_lock, flags);
> - hlist_del_rcu(&pid->pid_chain);
> + hlist_del_rcu(&tref->pid_chain);
> spin_unlock_irqrestore(&pidmap_lock, flags);
>
> - free_pidmap(&init_pid_ns, pid->nr);
> - call_rcu(&pid->rcu, delayed_put_pid);
> + free_pidmap(&init_pid_ns, tref->pid);
> + call_rcu(&tref->rcu, delayed_put_task_ref);
> }
>
> -struct pid *alloc_pid(void)
> +struct task_ref *alloc_task_ref(void)
> {
> - struct pid *pid;
> - enum pid_type type;
> + struct task_ref *tref;
> + enum task_ref_type type;
> int nr = -1;
>
> - pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
> - if (!pid)
> + tref = kmem_cache_alloc(task_ref_cachep, GFP_KERNEL);
> + if (!tref)
> goto out;
>
> nr = alloc_pidmap(current->nsproxy->pid_ns);
> if (nr < 0)
> goto out_free;

```

```

>
> - atomic_set(&pid->count, 1);
> - pid->nr = nr;
> - for (type = 0; type < PIDTYPE_MAX; ++type)
> - INIT_HLIST_HEAD(&pid->tasks[type]);
> + atomic_set(&tref->count, 1);
> + tref->pid = nr;
> + for (type = 0; type < REFTYPE_MAX; ++type)
> + INIT_HLIST_HEAD(&tref->tasks[type]);
>
> spin_lock_irq(&pidmap_lock);
> - hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(pid->nr)]);
> + hlist_add_head_rcu(&tref->pid_chain, &pid_hash[pid_hashfn(tref->pid)]);
> spin_unlock_irq(&pidmap_lock);
>
> out:
> - return pid;
> + return tref;
>
> out_free:
> - kmem_cache_free(pid_cachep, pid);
> - pid = NULL;
> + kmem_cache_free(task_ref_cachep, tref);
> + tref = NULL;
> goto out;
> }
>
> -struct pid *fastcall find_pid(int nr)
> +struct task_ref *fastcall find_task(int nr)
> {
> struct hlist_node *elem;
> - struct pid *pid;
> + struct task_ref *tref;
>
> - hlist_for_each_entry_rcu(pid, elem,
> + hlist_for_each_entry_rcu(tref, elem,
> &pid_hash[pid_hashfn(nr)], pid_chain) {
> - if (pid->nr == nr)
> - return pid;
> + if (tref->pid == nr)
> + return tref;
> }
> return NULL;
> }
> -EXPORT_SYMBOL_GPL(find_pid);
> +EXPORT_SYMBOL_GPL(find_task);
>
> /*

```

```

> - * attach_pid() must be called with the tasklist_lock write-held.
> + * attach_task_ref() must be called with the tasklist_lock write-held.
> */
> -int fastcall attach_pid(struct task_struct *task, enum pid_type type,
> - struct pid *pid)
> +int fastcall attach_task_ref(struct task_struct *task, enum task_ref_type type,
> + struct task_ref *tref)
> {
>   struct pid_link *link;
>
>   link = &task->pids[type];
> - link->pid = pid;
> - hlist_add_head_rcu(&link->node, &pid->tasks[type]);
> + link->tref = tref;
> + hlist_add_head_rcu(&link->node, &tref->tasks[type]);
>
>   return 0;
> }
>
> -void fastcall detach_pid(struct task_struct *task, enum pid_type type)
> +void fastcall detach_task_ref(struct task_struct *task, enum task_ref_type type)
> {
>   struct pid_link *link;
> - struct pid *pid;
> + struct task_ref *tref;
>   int tmp;
>
>   link = &task->pids[type];
> - pid = link->pid;
> + tref = link->tref;
>
>   hlist_del_rcu(&link->node);
> - link->pid = NULL;
> + link->tref = NULL;
>
> - for (tmp = PIDTYPE_MAX; --tmp >= 0; )
> -   if (!hlist_empty(&pid->tasks[tmp]))
> + for (tmp = REFTYPE_MAX; --tmp >= 0; )
> +   if (!hlist_empty(&tref->tasks[tmp]))
>   return;
>
> - free_pid(pid);
> + free_task_ref(tref);
> }
>
> -/* transfer_pid is an optimization of attach_pid(new), detach_pid(old) */
> -void fastcall transfer_pid(struct task_struct *old, struct task_struct *new,
> - enum pid_type type)

```

```

> +/* transfer_task_ref is an optimization of attach_pid(new), detach_pid(old) */
> +void fastcall transfer_task_ref(struct task_struct *old,
> +    struct task_struct *new,
> +    enum task_ref_type type)
> {
> - new->pids[type].pid = old->pids[type].pid;
> + new->pids[type].tref = old->pids[type].tref;
>   hlist_replace_rcu(&old->pids[type].node, &new->pids[type].node);
> - old->pids[type].pid = NULL;
> + old->pids[type].tref = NULL;
> }
>
> -struct task_struct * fastcall pid_task(struct pid *pid, enum pid_type type)
> +struct task_struct * fastcall pid_task(struct task_ref *tref, enum task_ref_type type)
> {
>   struct task_struct *result = NULL;
> - if (pid) {
> + if (tref) {
>   struct hlist_node *first;
> - first = rcu_dereference(pid->tasks[type].first);
> + first = rcu_dereference(tref->tasks[type].first);
>   if (first)
>     result = hlist_entry(first, struct task_struct, pids[(type)].node);
> }
> @@ -307,61 +308,61 @@ struct task_struct * fastcall pid_task(s
> /*
>  * Must be called under rcu_read_lock() or with tasklist_lock read-held.
>  */
> -struct task_struct *find_task_by_pid_type(int type, int nr)
> +struct task_struct *find_task_by_ref_type(int type, int nr)
> {
> - return pid_task(find_pid(nr), type);
> + return pid_task(find_task(nr), type);
> }
>
> -EXPORT_SYMBOL(find_task_by_pid_type);
> +EXPORT_SYMBOL(find_task_by_ref_type);
>
> -struct pid *get_task_pid(struct task_struct *task, enum pid_type type)
> +struct task_ref *get_task_ref(struct task_struct *task, enum task_ref_type type)
> {
> - struct pid *pid;
> + struct task_ref *tref;
>   rcu_read_lock();
> - pid = get_pid(task->pids[type].pid);
> + tref = get_pid(task->pids[type].tref);
>   rcu_read_unlock();
> - return pid;

```

```

> + return tref;
> }
>
> -struct task_struct *fastcall get_pid_task(struct pid *pid, enum pid_type type)
> +struct task_struct *fastcall get_pid_task(struct task_ref *tref, enum task_ref_type type)
> {
>     struct task_struct *result;
>     rcu_read_lock();
> - result = pid_task(pid, type);
> + result = pid_task(tref, type);
>     if (result)
>         get_task_struct(result);
>     rcu_read_unlock();
>     return result;
> }
>
> -struct pid *find_get_pid(pid_t nr)
> +struct task_ref *find_get_pid(pid_t nr)
> {
> - struct pid *pid;
> + struct task_ref *tref;
>
>     rcu_read_lock();
> - pid = get_pid(find_pid(nr));
> + tref = get_pid(find_task(nr));
>     rcu_read_unlock();
>
> - return pid;
> + return tref;
> }
>
> /*
>  * Used by proc to find the first pid that is greater then or equal to nr.
>  *
> - * If there is a pid at nr this function is exactly the same as find_pid.
> + * If there is a pid at nr this function is exactly the same as find_task.
>  */
> -struct pid *find_ge_pid(int nr)
> +struct task_ref *find_ge_pid(int nr)
> {
> - struct pid *pid;
> + struct task_ref *tref;
>
>     do {
> - pid = find_pid(nr);
> - if (pid)
> + tref = find_task(nr);
> + if (tref)

```

```

> break;
> nr = next_pidmap(current->nsproxy->pid_ns, nr);
> } while (nr > 0);
>
> - return pid;
> + return tref;
> }
> EXPORT_SYMBOL_GPL(find_get_pid);
>
> @@ -412,7 +413,8 @@ void __init pidmap_init(void)
> set_bit(0, init_pid_ns.pidmap[0].page);
> atomic_dec(&init_pid_ns.pidmap[0].nr_free);
>
> - pid_cachep = kmem_cache_create("pid", sizeof(struct pid),
> - __alignof__(struct pid),
> + task_ref_cachep = kmem_cache_create("task_ref",
> + sizeof(struct task_ref),
> + __alignof__(struct task_ref),
> SLAB_PANIC, NULL, NULL);
> }
> diff -puN kernel/rtmutex-debug.c~rename-struct-pid kernel/rtmutex-debug.c
> --- lxc/kernel/rtmutex-debug.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/rtmutex-debug.c 2007-04-10 16:18:30.000000000 -0700
> @@ -238,4 +238,3 @@ rt_mutex_deadlock_account_lock(struct rt
> void rt_mutex_deadlock_account_unlock(struct task_struct *task)
> {
> }
> -
> diff -puN kernel/signal.c~rename-struct-pid kernel/signal.c
> --- lxc/kernel/signal.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/signal.c 2007-04-10 16:18:30.000000000 -0700
> @@ -1224,22 +1224,22 @@ int group_send_sig_info(int sig, struct
> * control characters do (^C, ^Z etc)
> */
>
> -int __kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp)
> +int __kill_pgrp_info(int sig, struct siginfo *info, struct task_ref *pgrp)
> {
> struct task_struct *p = NULL;
> int retval, success;
>
> success = 0;
> retval = -ESRCH;
> - do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> + do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
> int err = group_send_sig_info(sig, info, p);
> success |= !err;
> retval = err;

```

```

> - } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, p);
> return success ? 0 : retval;
> }
>
> -int kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp)
> +int kill_pgrp_info(int sig, struct siginfo *info, struct task_ref *pgrp)
> {
>     int retval;
>
> @@ -1250,7 +1250,7 @@ int kill_pgrp_info(int sig, struct sigin
>     return retval;
> }
>
> -int kill_pid_info(int sig, struct siginfo *info, struct pid *pid)
> +int kill_pid_info(int sig, struct siginfo *info, struct task_ref *pid)
> {
>     int error;
>     struct task_struct *p;
> @@ -1259,7 +1259,7 @@ int kill_pid_info(int sig, struct siginf
>     if (unlikely(sig_needs_tasklist(sig)))
>         read_lock(&tasklist_lock);
>
> - p = pid_task(pid, PIDTYPE_PID);
> + p = pid_task(pid, REFTYPE_PID);
>     error = -ESRCH;
>     if (p)
>         error = group_send_sig_info(sig, info, p);
> @@ -1275,13 +1275,13 @@ kill_proc_info(int sig, struct siginfo *
> {
>     int error;
>     rcu_read_lock();
> - error = kill_pid_info(sig, info, find_pid(pid));
> + error = kill_pid_info(sig, info, find_task(pid));
>     rcu_read_unlock();
>     return error;
> }
>
> /* like kill_pid_info(), but doesn't use uid/euid of "current" */
> -int kill_pid_info_as_uid(int sig, struct siginfo *info, struct pid *pid,
> +int kill_pid_info_as_uid(int sig, struct siginfo *info, struct task_ref *pid,
>     uid_t uid, uid_t euid, u32 secid)
> {
>     int ret = -EINVAL;
> @@ -1291,7 +1291,7 @@ int kill_pid_info_as_uid(int sig, struct
>     return ret;
>
>     read_lock(&tasklist_lock);

```



```

> - p = pid_task(pid, PIDTYPE_PID);
> + p = pid_task(pid, REFTYPE_PID);
> if (!p) {
>     ret = -ESRCH;
>     goto out_unlock;
> @@ -1346,9 +1346,9 @@ static int kill_something_info(int sig,
>     read_unlock(&tasklist_lock);
>     ret = count ? retval : -ESRCH;
> } else if (pid < 0) {
> - ret = kill_pgrp_info(sig, info, find_pid(-pid));
> + ret = kill_pgrp_info(sig, info, find_task(-pid));
> } else {
> - ret = kill_pid_info(sig, info, find_pid(pid));
> + ret = kill_pid_info(sig, info, find_task(pid));
> }
> rcu_read_unlock();
> return ret;
> @@ -1437,13 +1437,13 @@ force_sigsegv(int sig, struct task_struc
> return 0;
> }
>
> -int kill_pgrp(struct pid *pid, int sig, int priv)
> +int kill_pgrp(struct task_ref *pid, int sig, int priv)
> {
>     return kill_pgrp_info(sig, __si_special(priv), pid);
> }
> EXPORT_SYMBOL(kill_pgrp);
>
> -int kill_pid(struct pid *pid, int sig, int priv)
> +int kill_pid(struct task_ref *pid, int sig, int priv)
> {
>     return kill_pid_info(sig, __si_special(priv), pid);
> }
> diff -puN kernel/sys.c~rename-struct-pid kernel/sys.c
> --- lxc/kernel/sys.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/sys.c 2007-04-10 16:18:30.000000000 -0700
> @@ -93,8 +93,8 @@ EXPORT_SYMBOL(fs_overflowgid);
> */
>
> int C_A_D = 1;
> -struct pid *cad_pid;
> -EXPORT_SYMBOL(cad_pid);
> +struct task_ref *cad_tref;
> +EXPORT_SYMBOL(cad_tref);
>
> /*
>  * Notifier list for kernel code which wants to be called
> @@ -657,7 +657,7 @@ asmlinkage long sys_setpriority(int whic

```

```

> struct task_struct *g, *p;
> struct user_struct *user;
> int error = -EINVAL;
> - struct pid *pgrp;
> + struct task_ref *pgrp;
>
> if (which > 2 || which < 0)
> goto out;
> @@ -681,12 +681,12 @@ asmlinkage long sys_setpriority(int whic
> break;
> case PRIO_PGRP:
> if (who)
> - pgrp = find_pid(who);
> + pgrp = find_task(who);
> else
> pgrp = task_pgrp(current);
> - do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> + do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
> error = set_one_prio(p, niceval, error);
> - } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, p);
> break;
> case PRIO_USER:
> user = current->user;
> @@ -721,7 +721,7 @@ asmlinkage long sys_getpriority(int whic
> struct task_struct *g, *p;
> struct user_struct *user;
> long niceval, retval = -ESRCH;
> - struct pid *pgrp;
> + struct task_ref *pgrp;
>
> if (which > 2 || which < 0)
> return -EINVAL;
> @@ -741,14 +741,14 @@ asmlinkage long sys_getpriority(int whic
> break;
> case PRIO_PGRP:
> if (who)
> - pgrp = find_pid(who);
> + pgrp = find_task(who);
> else
> pgrp = task_pgrp(current);
> - do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
> + do_each_referenced_task(pgrp, REFTYPE_PGID, p) {
> niceval = 20 - task_nice(p);
> if (niceval > retval)
> retval = niceval;
> - } while_each_pid_task(pgrp, PIDTYPE_PGID, p);
> + } while_each_referenced_task(pgrp, REFTYPE_PGID, p);

```

```

> break;
> case PRIO_USER:
> user = current->user;
> @@ -1474,7 +1474,7 @@ asmlinkage long sys_setpgid(pid_t pid, p
>
> if (pgid != pid) {
> struct task_struct *g =
> - find_task_by_pid_type(PIDTYPE_PGID, pgid);
> + find_task_by_ref_type(REFTYPE_PGID, pgid);
>
> if (!g || task_session(g) != task_session(group_leader))
> goto out;
> @@ -1485,9 +1485,9 @@ asmlinkage long sys_setpgid(pid_t pid, p
> goto out;
>
> if (process_group(p) != pgid) {
> - detach_pid(p, PIDTYPE_PGID);
> + detach_task_ref(p, REFTYPE_PGID);
> p->signal->pgrp = pgid;
> - attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
> + attach_task_ref(p, REFTYPE_PGID, find_task(pgid));
> }
>
> err = 0;
> @@ -1571,7 +1571,7 @@ asmlinkage long sys_setsid(void)
> * session id and so the check will always fail and make it so
> * init cannot successfully call setsid.
> */
> - if (session > 1 && find_task_by_pid_type(PIDTYPE_PGID, session))
> + if (session > 1 && find_task_by_ref_type(REFTYPE_PGID, session))
> goto out;
>
> group_leader->signal->leader = 1;
> diff -puN kernel/sysctl.c~rename-struct-pid kernel/sysctl.c
> --- lxc/kernel/sysctl.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/kernel/sysctl.c 2007-04-10 16:18:30.000000000 -0700
> @@ -2127,22 +2127,22 @@ int proc_dointvec_ms_jiffies(ctl_table *
> static int proc_do_cad_pid(ctl_table *table, int write, struct file *filp,
> void __user *buffer, size_t *lenp, loff_t *ppos)
> {
> - struct pid *new_pid;
> + struct task_ref *new_tref;
> pid_t tmp;
> int r;
>
> - tmp = pid_nr(cad_pid);
> + tmp = tref_to_pid(cad_tref);
>

```

```

> r = __do_proc_dointvec(&tmp, table, write, filp, buffer,
>      lenp, ppos, NULL, NULL);
> if (r || !write)
>     return r;
>
> - new_pid = find_get_pid(tmp);
> - if (!new_pid)
> + new_tref = find_get_pid(tmp);
> + if (!new_tref)
>     return -ESRCH;
>
> - put_pid(xchg(&cad_pid, new_pid));
> + put_task_ref(xchg(&cad_tref, new_tref));
>     return 0;
> }
>
> diff -puN mm/mempolicy.c~rename-struct-pid mm/mempolicy.c
> --- lxc/mm/mempolicy.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/mm/mempolicy.c 2007-04-10 16:18:30.000000000 -0700
> @@ -923,7 +923,7 @@ asmlinkage long sys_migrate_pages(pid_t
>
> /* Find the mm_struct */
> read_lock(&tasklist_lock);
> - task = pid ? find_task_by_pid(pid) : current;
> + task = pid ? find_task_by_ref(pid) : current;
> if (!task) {
>     read_unlock(&tasklist_lock);
>     return -ESRCH;
> @@ -1909,4 +1909,3 @@ out:
>     m->version = (vma != priv->tail_vma) ? vma->vm_start : 0;
>     return 0;
> }
> -
> diff -puN mm/migrate.c~rename-struct-pid mm/migrate.c
> --- lxc/mm/migrate.c~rename-struct-pid 2007-04-10 16:18:30.000000000 -0700
> +++ lxc-dave/mm/migrate.c 2007-04-10 16:18:30.000000000 -0700
> @@ -887,7 +887,7 @@ asmlinkage long sys_move_pages(pid_t pid
>
> /* Find the mm_struct */
> read_lock(&tasklist_lock);
> - task = pid ? find_task_by_pid(pid) : current;
> + task = pid ? find_task_by_ref(pid) : current;
> if (!task) {
>     read_unlock(&tasklist_lock);
>     return -ESRCH;
> }
>
>

```

> Containers mailing list

> Containers@lists.linux-foundation.org
> <https://lists.linux-foundation.org/mailman/listinfo/containers>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
