
Subject: [patch -mm 2/4] rename struct pspace to struct pid_namespace

Posted by [Cedric Le Goater](#) on Fri, 08 Sep 2006 17:00:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

Rename struct pspace to struct pid_namespace for consistency with other namespaces (uts_namespace and ipc_namespace). Also rename include/linux/pspace.h to include/linux/pid_namespace.h and variables from pspace to pid_ns.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

Cc: Dave Hansen <haveblue@us.ibm.com>

Cc: Serge Hallyn <serue@us.ibm.com>

```
fs/proc/proc_misc.c      | 4 +--
include/linux/pid_namespace.h | 23 ++++++
include/linux/pspace.h    | 23 -----
kernel/pid.c              | 49 ++++++-----
4 files changed, 50 insertions(+), 49 deletions(-)
```

Index: 2.6.18-rc6-mm1/include/linux/pid_namespace.h

```
=====
--- /dev/null
+++ 2.6.18-rc6-mm1/include/linux/pid_namespace.h
@@ -0,0 +1,23 @@
+#ifndef _LINUX_PID_NS_H
+#define _LINUX_PID_NS_H
+
+#include <linux/sched.h>
+#include <linux/mm.h>
+#include <linux/threads.h>
+#include <linux/pid.h>
+
+struct pidmap {
+    atomic_t nr_free;
+    void *page;
+};
+
+#define PIDMAP_ENTRIES    ((PID_MAX_LIMIT + 8*PAGE_SIZE - 1)/PAGE_SIZE/8)
+
+struct pid_namespace {
+    struct pidmap pidmap[PIDMAP_ENTRIES];
+    int last_pid;
+};
+
+extern struct pid_namespace init_pid_ns;
+
+#endif /* _LINUX_PID_NS_H */
```

Index: 2.6.18-rc6-mm1/include/linux/pspace.h

--- 2.6.18-rc6-mm1.orig/include/linux/pspace.h

+++ /dev/null

@@ -1,23 +0,0 @@

-#ifndef _LINUX_PSPACE_H

-#define _LINUX_PSPACE_H

-#include <linux/sched.h>

-#include <linux/mm.h>

-#include <linux/threads.h>

-#include <linux/pid.h>

-struct pidmap {

- atomic_t nr_free;

- void *page;

-};

-#define PIDMAP_ENTRIES ((PID_MAX_LIMIT + 8*PAGE_SIZE - 1)/PAGE_SIZE/8)

-struct pspace {

- struct pidmap pidmap[PIDMAP_ENTRIES];

- int last_pid;

-};

-extern struct pspace init_pspace;

-#endif /* _LINUX_PSPACE_H */

Index: 2.6.18-rc6-mm1/kernel/pid.c

--- 2.6.18-rc6-mm1.orig/kernel/pid.c

+++ 2.6.18-rc6-mm1/kernel/pid.c

@@ -26,7 +26,7 @@

#include <linux/init.h>

#include <linux/bootmem.h>

#include <linux/hash.h>

-#include <linux/pspace.h>

+#include <linux/pid_namespace.h>

#define pid_hashfn(nr) hash_long((unsigned long)nr, pidhash_shift)

static struct hlist_head *pid_hash;

@@ -43,9 +43,10 @@ int pid_max_max = PID_MAX_LIMIT;

#define BITS_PER_PAGE (PAGE_SIZE*8)

#define BITS_PER_PAGE_MASK (BITS_PER_PAGE-1)

-static inline int mk_pid(struct pspace *pspace, struct pidmap *map, int off)

+static inline int mk_pid(struct pid_namespace *pid_ns,

+ struct pidmap *map, int off)

```

{
- return (map - pspace->pidmap)*BITS_PER_PAGE + off;
+ return (map - pid_ns->pidmap)*BITS_PER_PAGE + off;
}

#define find_next_offset(map, off) \
@@ -57,7 +58,7 @@ static inline int mk_pid(struct pspace *
 * value does not cause lots of bitmaps to be allocated, but
 * the scheme scales to up to 4 million PIDs, runtime.
 */
-struct pspace init_pspace = {
+struct pid_namespace init_pid_ns = {
    .pidmap = {
        [ 0 ... PIDMAP_ENTRIES-1] = { ATOMIC_INIT(BITS_PER_PAGE), NULL }
    },
@@ -80,25 +81,25 @@ struct pspace init_pspace = {

static __cacheline_aligned_in_smp DEFINE_SPINLOCK(pidmap_lock);

-static fastcall void free_pidmap(struct pspace *pspace, int pid)
+static fastcall void free_pidmap(struct pid_namespace *pid_ns, int pid)
{
- struct pidmap *map = pspace->pidmap + pid / BITS_PER_PAGE;
+ struct pidmap *map = pid_ns->pidmap + pid / BITS_PER_PAGE;
    int offset = pid & BITS_PER_PAGE_MASK;

    clear_bit(offset, map->page);
    atomic_inc(&map->nr_free);
}

-static int alloc_pidmap(struct pspace *pspace)
+static int alloc_pidmap(struct pid_namespace *pid_ns)
{
- int i, offset, max_scan, pid, last = pspace->last_pid;
+ int i, offset, max_scan, pid, last = pid_ns->last_pid;
    struct pidmap *map;

    pid = last + 1;
    if (pid >= pid_max)
        pid = RESERVED_PIDS;
    offset = pid & BITS_PER_PAGE_MASK;
- map = &pspace->pidmap[pid/BITS_PER_PAGE];
+ map = &pid_ns->pidmap[pid/BITS_PER_PAGE];
    max_scan = (pid_max + BITS_PER_PAGE - 1)/BITS_PER_PAGE - !offset;
    for (i = 0; i <= max_scan; ++i) {
        if (unlikely(!map->page)) {
@@ -120,11 +121,11 @@ static int alloc_pidmap(struct pspace *p
    do {

```

```

    if (!test_and_set_bit(offset, map->page)) {
        atomic_dec(&map->nr_free);
-   pspace->last_pid = pid;
+   pid_ns->last_pid = pid;
        return pid;
    }
    offset = find_next_offset(map, offset);
-   pid = mk_pid(pspace, map, offset);
+   pid = mk_pid(pid_ns, map, offset);
/*
 * find_next_offset() found a bit, the pid from it
 * is in-bounds, and if we fell back to the last
@@ -135,34 +136,34 @@ static int alloc_pidmap(struct pspace *p
    (i != max_scan || pid < last ||
     !((last+1) & BITS_PER_PAGE_MASK));
}
- if (map < &pspace->pidmap[(pid_max-1)/BITS_PER_PAGE]) {
+ if (map < &pid_ns->pidmap[(pid_max-1)/BITS_PER_PAGE]) {
    ++map;
    offset = 0;
} else {
-   map = &pspace->pidmap[0];
+   map = &pid_ns->pidmap[0];
    offset = RESERVED_PIDS;
    if (unlikely(last == offset))
        break;
}
-   pid = mk_pid(pspace, map, offset);
+   pid = mk_pid(pid_ns, map, offset);
}
return -1;
}

-static int next_pidmap(struct pspace *pspace, int last)
+static int next_pidmap(struct pid_namespace *pid_ns, int last)
{
    int offset;
    struct pidmap *map, *end;

    offset = (last + 1) & BITS_PER_PAGE_MASK;
-   map = &pspace->pidmap[(last + 1)/BITS_PER_PAGE];
-   end = &pspace->pidmap[PIDMAP_ENTRIES];
+   map = &pid_ns->pidmap[(last + 1)/BITS_PER_PAGE];
+   end = &pid_ns->pidmap[PIDMAP_ENTRIES];
    for (; map < end; map++, offset = 0) {
        if (unlikely(!map->page))
            continue;
        offset = find_next_bit((map)->page, BITS_PER_PAGE, offset);
    }
}

```

```

    if (offset < BITS_PER_PAGE)
-   return mk_pid(pspace, map, offset);
+   return mk_pid(pid_ns, map, offset);
    }
    return -1;
}

@@ -192,7 +193,7 @@ fastcall void free_pid(struct pid *pid)
    hlist_del_rcu(&pid->pid_chain);
    spin_unlock_irqrestore(&pidmap_lock, flags);

-   free_pidmap(&init_pspace, pid->nr);
+   free_pidmap(&init_pid_ns, pid->nr);
    call_rcu(&pid->rcu, delayed_put_pid);
}

@@ -206,7 +207,7 @@ struct pid *alloc_pid(void)
    if (!pid)
        goto out;

-   nr = alloc_pidmap(&init_pspace);
+   nr = alloc_pidmap(&init_pid_ns);
    if (nr < 0)
        goto out_free;

@@ -340,7 +341,7 @@ struct pid *find_ge_pid(int nr)
    pid = find_pid(nr);
    if (pid)
        break;
-   nr = next_pidmap(&init_pspace, nr);
+   nr = next_pidmap(&init_pid_ns, nr);
} while (nr > 0);

    return pid;
@@ -374,10 +375,10 @@ void __init pidhash_init(void)

void __init pidmap_init(void)
{
-   init_pspace.pidmap[0].page = kzalloc(PAGE_SIZE, GFP_KERNEL);
+   init_pid_ns.pidmap[0].page = kzalloc(PAGE_SIZE, GFP_KERNEL);
    /* Reserve PID 0. We never call free_pidmap(0) */
-   set_bit(0, init_pspace.pidmap[0].page);
-   atomic_dec(&init_pspace.pidmap[0].nr_free);
+   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),
Index: 2.6.18-rc6-mm1/fs/proc/proc_misc.c

```

=====

```
--- 2.6.18-rc6-mm1.orig/fs/proc/proc_misc.c
+++ 2.6.18-rc6-mm1/fs/proc/proc_misc.c
@@ -45,7 +45,7 @@
#include <linux/sysrq.h>
#include <linux/vmalloc.h>
#include <linux/crash_dump.h>
-#include <linux/pspace.h>
+#include <linux/pid_namespace.h>
#include <asm/uaccess.h>
#include <asm/pgtable.h>
#include <asm/io.h>
@@ -92,7 +92,7 @@ static int loadavg_read_proc(char *page,
    LOAD_INT(a), LOAD_FRAC(a),
    LOAD_INT(b), LOAD_FRAC(b),
    LOAD_INT(c), LOAD_FRAC(c),
- nr_running(), nr_threads, init_pspace.last_pid);
+ nr_running(), nr_threads, init_pid_ns.last_pid);
    return proc_calc_metrics(page, start, off, count, eof, len);
}
```

--

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
