Subject: [RFC][PATCH 7/7] UBC: proc interface
Posted by dev on Wed, 16 Aug 2006 15:42:28 GMT
View Forum Message <> Reply to Message

Add proc interface (/proc/user_beancounters) allowing to see current
state (usage/limits/fails for each UB). Implemented via seq files.

Signed-Off-By: Pavel Emelianov <xemul@sw.ru>
Signed-Off-By: Kirill Korotaev <dev@sw.ru>

---
 init/main.c        |    1
 kernel/ub/Makefile |    1
 kernel/ub/proc.c   |  205 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 207 insertions(+)

--- ./init/main.c.ubproc 2006-07-31 18:40:20.000000000 +0400
+++ ./init/main.c 2006-08-03 16:02:19.000000000 +0400
@@ -578,6 +578,7 @@ asmlinkage void __init start_kernel(void
  page_writeback_init();
 #ifdef CONFIG_PROC_FS
  proc_root_init();
+ ub_init_proc();
 #endif
  cpuset_init();
  taskstats_init_early();
--- ./kernel/ub/Makefile.ubproc 2006-07-31 17:49:05.000000000 +0400
+++ ./kernel/ub/Makefile 2006-08-01 11:08:39.000000000 +0400
@@ -4,3 +4,4 @@ obj-$(CONFIG_USER_RESOURCE) += beancount
 obj-$(CONFIG_USER_RESOURCE) += misc.o
 obj-y += sys.o
 obj-$(CONFIG_USER_RESOURCE) += kmem.o
+obj-$(CONFIG_USER_RESOURCE) += proc.o
--- ./kernel/ub/proc.c.ubproc 2006-08-01 10:22:09.000000000 +0400
+++ ./kernel/ub/proc.c 2006-08-03 15:50:35.000000000 +0400
@@ -0,0 +1,205 @@
+/*
+ *  kernel/ub/proc.c
+ *
+ *  Copyright (C) 2006 OpenVZ. SWsoft Inc.
+ *
+ */
+
+#include <linux/sched.h>
+#include <linux/kernel.h>
+#include <linux/proc_fs.h>
+#include <linux/seq_file.h>
+

```
+#include <ub/beancounter.h>
+
+#ifdef CONFIG_PROC_FS
+
+#if BITS_PER_LONG == 32
+static const char *head_fmt = "%10s %-12s %10s %10s %10s %10s %10s\n";
+static const char *res_fmt = "%10s %-12s %10lu %10lu %10lu %10lu %10lu\n";
+#else
+static const char *head_fmt = "%10s %-12s %20s %20s %20s %20s %20s\n";
+static const char *res_fmt = "%10s %-12s %20lu %20lu %20lu %20lu %20lu\n";
+#endif
+
+static void ub_show_header(struct seq_file *f)
+{
+ seq_printf(f, head_fmt, "uid", "resource",
+   "held", "maxheld", "barrier", "limit", "failcnt");
+}
+
+static void ub_show_res(struct seq_file *f, struct user_beancounter *ub, int r)
+{
+ char ub_uid[64];
+
+ if (r == 0)
+  ub_print_uid(ub, ub_uid, sizeof(ub_uid));
+ else
+  strcpy(ub_uid, "");
+
+ seq_printf(f, res_fmt, ub_uid, ub_rnames[r],
+   ub->ub_parms[r].held,
+   ub->ub_parms[r].maxheld,
+   ub->ub_parms[r].barrier,
+   ub->ub_parms[r].limit,
+   ub->ub_parms[r].failcnt);
+}
+
+static struct ub_seq_struct {
+ unsigned long flags;
+ int slot;
+ struct user_beancounter *ub;
+} ub_seq_ctx;
+
+static int ub_show(struct seq_file *f, void *v)
+{
+ int res;
+
+ for (res = 0; res < UB_RESOURCES; res++)
+  ub_show_res(f, ub_seq_ctx.ub, res);
+ return 0;
```

```
+}
+
+static void *ub_start_ctx(struct seq_file *f, unsigned long p, int sub)
+{
+ struct user_beancounter *ub;
+ struct hlist_node *pos;
+ unsigned long flags;
+ int slot;
+
+ if (p == 0)
+  ub_show_header(f);
+
+ spin_lock_irqsave(&ub_hash_lock, flags);
+ ub_seq_ctx.flags = flags;
+
+ for (slot = 0; slot < UB_HASH_SIZE; slot++)
+  hlist_for_each_entry (ub, pos, &ub_hash[slot], hash) {
+   if (!sub && ub->parent != NULL)
+    continue;
+
+   if (p-- == 0) {
+    ub_seq_ctx.ub = ub;
+    ub_seq_ctx.slot = slot;
+    return &ub_seq_ctx;
+   }
+  }
+
+ return NULL;
+}
+
+static void *ub_next_ctx(struct seq_file *f, loff_t *ppos, int sub)
+{
+ struct user_beancounter *ub;
+ struct hlist_node *pos;
+ int slot;
+
+ ub = ub_seq_ctx.ub;
+
+ pos = &ub->hash;
+ hlist_for_each_entry_continue (ub, pos, hash) {
+  if (!sub && ub->parent != NULL)
+   continue;
+
+  ub_seq_ctx.ub = ub;
+  (*ppos)++;
+  return &ub_seq_ctx;
+ }
+
```

```
+ for (slot = ub_seq_ctx.slot + 1; slot < UB_HASH_SIZE; slot++)
+  hlist_for_each_entry (ub, pos, &ub_hash[slot], hash) {
+   if (!sub && ub->parent != NULL)
+    continue;
+
+   ub_seq_ctx.ub = ub;
+   ub_seq_ctx.slot = slot;
+   (*ppos)++;
+   return &ub_seq_ctx;
+  }
+
+ return NULL;
+}
+
+static void *ub_start(struct seq_file *f, loff_t *ppos)
+{
+ return ub_start_ctx(f, *ppos, 0);
+}
+
+static void *ub_sub_start(struct seq_file *f, loff_t *ppos)
+{
+ return ub_start_ctx(f, *ppos, 1);
+}
+
+static void *ub_next(struct seq_file *f, void *v, loff_t *pos)
+{
+ return ub_next_ctx(f, pos, 0);
+}
+
+static void *ub_sub_next(struct seq_file *f, void *v, loff_t *pos)
+{
+ return ub_next_ctx(f, pos, 1);
+}
+
+static void ub_stop(struct seq_file *f, void *v)
+{
+ unsigned long flags;
+
+ flags = ub_seq_ctx.flags;
+ spin_unlock_irqrestore(&ub_hash_lock, flags);
+}
+
+static struct seq_operations ub_seq_ops = {
+ .start = ub_start,
+ .next  = ub_next,
+ .stop  = ub_stop,
+ .show  = ub_show
+};
```

```
+
+static int ub_open(struct inode *inode, struct file *filp)
+{
+ return seq_open(filp, &ub_seq_ops);
+}
+
+static struct file_operations ub_file_operations = {
+ .open  = ub_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = seq_release,
+};
+
+static struct seq_operations ub_sub_seq_ops = {
+ .start = ub_sub_start,
+ .next  = ub_sub_next,
+ .stop  = ub_stop,
+ .show  = ub_show
+};
+
+static int ub_sub_open(struct inode *inode, struct file *filp)
+{
+ return seq_open(filp, &ub_sub_seq_ops);
+}
+
+static struct file_operations ub_sub_file_operations = {
+ .open  = ub_sub_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = seq_release,
+};
+
+void __init ub_init_proc(void)
+{
+ struct proc_dir_entry *entry;
+
+ entry = create_proc_entry("user_beancounters", S_IRUGO, NULL);
+ if (entry)
+  entry->proc_fops = &ub_file_operations;
+ else
+  panic("Can't create /proc/user_beancounters\n");
+
+ entry = create_proc_entry("user_beancounters_sub", S_IRUGO, NULL);
+ if (entry)
+  entry->proc_fops = &ub_sub_file_operations;
+ else
+  panic("Can't create /proc/user_beancounters_sub\n");
+}
```

```
+#endif
```