

[patch 2.6.12-rc1-mm4] fork_connector: add a fork connector

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-03/9719.html>

From: Guillaume Thouvenin (guillaume.thouvenin_at_bull.net)

Date: 03/31/05

To: Andrew Morton <akpm@osdl.org>, Greg KH <greg@kroah.com>

Date: Thu, 31 Mar 2005 15:59:02 +0200

This patch adds a fork connector in the `do_fork()` routine. It sends a netlink datagram when enabled. The message can be read by a user space application. By this way, the user space application is alerted when a fork occurs.

It uses the userspace <-> kernelspace connector that works on top of the netlink protocol. The fork connector is enabled or disabled by sending a message to the connector. The unique sequence number of messages can be used to check if a message is lost. In this patch we use a `cn_fork_msg` structure (thanks to Dean Gaudet) rather than zeroing 64 bytes of memory and doing conversions to decimal ascii as done before. Also we move the `fork_connector()` inline code in the header file `include/linux/connector.h` as suggested by Paul Jackson (thanks).

The fork connector is used by the Enhanced Linux System Accounting project <http://elsa.sourceforge.net>

When the fork connector is disabled, the "lat_proc fork" test returns a value equal to:

Process fork+exit: 150.2076 microseconds

When the fork connector is enabled, the same test returns:

Process fork+exit: 153.7778 microseconds

So the overhead (the construction and the sending of the message) is around 2%. If we use the CBUS patch <http://lkml.org/lkml/2005/3/20/14> instead of `cn_netlink_send()` routine we can reduce this overhead near to 0%. We're waiting for the inclusion of the CBUS in the -mm tree.

This patch applies to 2.6.12-rc1-mm4.

Signed-off-by: Guillaume Thouvenin <guillaume.thouvenin@bull.net>

`drivers/connector/Kconfig` | 11 +++++

Linux-Kernel: [patch 2.6.12-rc1-mm4] fork_connector: add a fork connector

```
drivers/connector/Makefile | 1
drivers/connector/cn_fork.c | 104 ++++++
include/linux/connector.h | 53 ++++++
kernel/fork.c | 3 +
5 files changed, 172 insertions(+)
```

Index: linux-2.6.12-rc1-mm4-cnfork/drivers/connector/Kconfig

=====

--- linux-2.6.12-rc1-mm4-cnfork.orig/drivers/connector/Kconfig 2005-03-31 14:56:02.000000000 +02

+++ linux-2.6.12-rc1-mm4-cnfork/drivers/connector/Kconfig 2005-03-31 14:57:52.000000000 +02

@@ -10,4 +10,15 @@ config CONNECTOR

Connector support can also be built as a module. If so, the module will be called cn.ko.

+config FORK_CONNECTOR

+ bool "Enable fork connector"

+ depends on CONNECTOR=y

+ default y

+ ---help---

+ It adds a connector in kernel/fork.c:do_fork() function. When a fork occurs, netlink is used to transfer information about the parent and its child. This information can be used by a user space application. The fork connector can be enable/disable by sending a message to the connector with the corresponding group id.

+ endmenu

Index: linux-2.6.12-rc1-mm4-cnfork/drivers/connector/Makefile

=====

--- linux-2.6.12-rc1-mm4-cnfork.orig/drivers/connector/Makefile 2005-03-31 14:56:02.000000000 +02

+++ linux-2.6.12-rc1-mm4-cnfork/drivers/connector/Makefile 2005-03-31 14:57:52.000000000 +02

@@ -1,2 +1,3 @@

obj-\$(CONFIG_CONNECTOR) += cn.o

+obj-\$(CONFIG_FORK_CONNECTOR) += cn_fork.o

cn-objs := cn_queue.o connector.o

Index: linux-2.6.12-rc1-mm4-cnfork/drivers/connector/cn_fork.c

=====

--- linux-2.6.12-rc1-mm4-cnfork.orig/drivers/connector/cn_fork.c 2003-01-30 11:24:37.000000

+++ linux-2.6.12-rc1-mm4-cnfork/drivers/connector/cn_fork.c 2005-03-31 14:57:52.000000000 +02

@@ -0,0 +1,104 @@

+/*

+ * cn_fork.c

+ *

+ * 2005 Copyright (c) Guillaume Thouvenin <guillaume.thouvenin@bull.net>

+ * All rights reserved.

+ *

+ * This program is free software; you can redistribute it and/or modify

+ * it under the terms of the GNU General Public License as published by

+ * the Free Software Foundation; either version 2 of the License, or

+ * (at your option) any later version.

+ *

+ * This program is distributed in the hope that it will be useful,

+ * but WITHOUT ANY WARRANTY; without even the implied warranty of

+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

+ * GNU General Public License for more details.

+ *

+ * You should have received a copy of the GNU General Public License

+ * along with this program; if not, write to the Free Software

+ * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

+ */

+

+#include <linux/module.h>

+#include <linux/kernel.h>

+#include <linux/init.h>

Linux-Kernel: [patch 2.6.12-rc1-mm4] fork_connector: add a fork connector

```
+
+#include <linux/connector.h>
+
+MODULE_LICENSE("GPL");
+MODULE_AUTHOR("Guillaume Thouvenin <guillaume.thouvenin@bull.net>");
+MODULE_DESCRIPTION("Enable or disable the usage of the fork connector");
+
+int cn_fork_enable = 0;
+struct cb_id cb_fork_id = { CN_IDX_FORK, CN_VAL_FORK };
+
+static inline void cn_fork_send_status(void)
+{
+    /* TODO */
+    printk(KERN_INFO "cn_fork_enable == %d\n", cn_fork_enable);
+}
+
+/**
+ * cn_fork_callback - enable or disable the fork connector
+ * @data: message send by the connector
+ *
+ * The callback allows to enable or disable the sending of information
+ * about fork in the do_fork() routine. To enable the fork, the user
+ * space application must send the integer 1 in the data part of the
+ * message. To disable the fork connector, it must send the integer 0.
+ */
+static void cn_fork_callback(void *data)
+{
+    struct cn_msg *msg = (struct cn_msg *)data;
+    int action;
+
+    if (cn_already_initialized && (msg->len == sizeof(cn_fork_enable))) {
+        memcpy(&action, msg->data, sizeof(cn_fork_enable));
+        switch(action) {
+            case FORK_CN_START:
+                cn_fork_enable = 1;
+                break;
+            case FORK_CN_STOP:
+                cn_fork_enable = 0;
+                break;
+            case FORK_CN_STATUS:
+                cn_fork_send_status();
+                break;
+        }
+    }
+}
+
+/**
+ * cn_fork_init - initialization entry point
+ *
+ * This routine will be run at kernel boot time because this driver is
+ * built in the kernel. It adds the connector callback to the connector
+ * driver.
+ */
+static int cn_fork_init(void)
+{
+    int err;
+
+    err = cn_add_callback(&cb_fork_id, "cn_fork", &cn_fork_callback);
+    if (err) {
+        printk(KERN_WARNING "Failed to register cn_fork\n");
+        return -EINVAL;
+    }
+}
```

Linux-Kernel: [patch 2.6.12-rc1-mm4] fork_connector: add a fork connector

```

+
+     printk(KERN_NOTICE "cn_fork is registered\n");
+     return 0;
+}
+
+/**
+ * cn_fork_exit - exit entry point
+ *
+ * As this driver is always statically compiled into the kernel the
+ * cn_fork_exit has no effect.
+ */
+static void cn_fork_exit(void)
+{
+     cn_del_callback(&cb_fork_id);
+}
+
+module_init(cn_fork_init);
+module_exit(cn_fork_exit);
Index: linux-2.6.12-rc1-mm4-cnfork/include/linux/connector.h
=====
--- linux-2.6.12-rc1-mm4-cnfork.orig/include/linux/connector.h    2005-03-31 14:56:05.000000000 +02
+++ linux-2.6.12-rc1-mm4-cnfork/include/linux/connector.h        2005-03-31 14:57:52.000000000 +02
@@ -28,10 +28,16 @@
     #define CN_VAL_KOBJECT_UEVENT                0x0000
     #define CN_IDX_SUPERIO                      0xaabb /* SuperIO subsystem */
     #define CN_VAL_SUPERIO                      0xccdd
+    #define CN_IDX_FORK                        0xfedd /* fork events */
+    #define CN_VAL_FORK                        0xbeef

     #define CONNECTOR_MAX_MSG_SIZE              1024

+    #define FORK_CN_STOP      0
+    #define FORK_CN_START    1
+    #define FORK_CN_STATUS   2
+
+    struct cb_id
+    {
+        __u32                idx;
@@ -64,6 +70,12 @@ struct cn_ctl_msg
+        __u8                data[0];
+    };

+struct cn_fork_msg
+{
+    int cpu;
+    int ppid;
+    int cpid;
+};

+    #ifdef __KERNEL__

@@ -133,6 +145,8 @@ struct cn_dev
+};

+extern int cn_already_initialized;
+extern int cn_fork_enable;
+extern struct cb_id cb_fork_id;

+int cn_add_callback(struct cb_id *, char *, void (* callback)(void *));
+void cn_del_callback(struct cb_id *);
@@ -146,5 +160,44 @@ void cn_queue_free_dev(struct cn_queue_d

```

Linux-Kernel: [patch 2.6.12-rc1-mm4] fork_connector: add a fork connector

```

int cn_cb_equal(struct cb_id *, struct cb_id *);

#ifdef CONFIG_FORK_CONNECTOR
+
#define CN_FORK_INFO_SIZE      sizeof(struct cn_fork_msg)
#define CN_FORK_MSG_SIZE      (sizeof(struct cn_msg) + CN_FORK_INFO_SIZE)
+
+static DEFINE_PER_CPU(unsigned long, fork_counts);
+
+static inline void fork_connector(pid_t parent, pid_t child)
+{
+    if (cn_fork_enable) {
+        struct cn_msg *msg;
+        struct cn_fork_msg *forkmsg;
+        __u8 buffer[CN_FORK_MSG_SIZE];
+
+        msg = (struct cn_msg *)buffer;
+
+        memcpy(&msg->id, &cb_fork_id, sizeof(msg->id));
+
+        msg->ack = 0; /* not used */
+        msg->seq = get_cpu_var(fork_counts)++;
+
+        msg->len = CN_FORK_INFO_SIZE;
+        forkmsg = (struct cn_fork_msg *)msg->data;
+        forkmsg->cpu = smp_processor_id();
+        forkmsg->ppid = parent;
+        forkmsg->cpid = child;
+
+        put_cpu_var(fork_counts);
+
+        cn_netlink_send(msg, CN_IDX_FORK);
+    }
+}
+
+static inline void fork_connector(pid_t parent, pid_t child)
+{
+    return;
+}
+
+endif /* CONFIG_FORK_CONNECTOR */
+
+endif /* __KERNEL__ */
+endif /* __CONNECTOR_H */
Index: linux-2.6.12-rc1-mm4-cnfork/kernel/fork.c
=====
--- linux-2.6.12-rc1-mm4-cnfork.orig/kernel/fork.c      2005-03-31 14:56:05.000000000 +0200
+++ linux-2.6.12-rc1-mm4-cnfork/kernel/fork.c      2005-03-31 14:57:52.000000000 +0200
@@ -41,6 +41,7 @@
#include <linux/profile.h>
#include <linux/rmap.h>
#include <linux/acct.h>
+#include <linux/connector.h>

#include <asm/pgtable.h>
#include <asm/pgalloc.h>
@@ -1249,6 +1250,8 @@ long do_fork(unsigned long clone_flags,
                if (unlikely(current->ptrace & PT_TRACE_VFORK_DONE))
                    ptrace_notify ((PT_TRACE_EVENT_VFORK_DONE << 8) | SIGTRAP);
            }
+
+        fork_connector(current->pid, p->pid);

```

Linux-Kernel: [patch 2.6.12-rc1-mm4] fork_connector: add a fork connector

```
} else {  
    free_pidmap(pid);  
    pid = PTR_ERR(p);  
-
```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>