
Subject: [patch i2o 6/6] i2o debug output cleanup
Posted by [vaverin](#) on Tue, 15 May 2007 12:48:16 GMT
[View Forum Message](#) <> [Reply to Message](#)

fixed output of i2o debug messages, extra KERN_ are removed

Signed-off-by: Vasily Averin <vvs@sw.ru>

```
--- lk2.6/drivers/message/i2o/debug.c
+++ lk2.6/drivers/message/i2o/debug.c
@@ -24,7 +24,7 @@ void i2o_report_status(const char *sever
 if (cmd == I2O_CMD_UTIL_EVT_REGISTER)
 return; // No status in this reply

- printk(KERN_DEBUG "%s%s: ", severity, str);
+ printk("%s%s: ", severity, str);

 if (cmd < 0x1F) // Utility cmd
 i2o_report_util_cmd(cmd);
@@ -32,7 +32,7 @@ void i2o_report_status(const char *sever
 else if (cmd >= 0xA0 && cmd <= 0xEF) // Executive cmd
 i2o_report_exec_cmd(cmd);
 else
- printk(KERN_DEBUG "Cmd = %0#2x, ", cmd); // Other cmds
+ printk("Cmd = %0#2x, ", cmd); // Other cmds

 if (msg[0] & MSG_FAIL) {
 i2o_report_fail_status(req_status, msg);
@@ -44,7 +44,7 @@ void i2o_report_status(const char *sever
 if (cmd < 0x1F || (cmd >= 0xA0 && cmd <= 0xEF))
 i2o_report_common_dsc(detailed_status);
 else
- printk(KERN_DEBUG " / DetailedStatus = %0#4x.\n",
+ printk(" / DetailedStatus = %0#4x.\n",
 detailed_status);
 }

@@ -89,10 +89,10 @@ static void i2o_report_fail_status(u8 re
};

 if (req_status == I2O_FSC_TRANSPORT_UNKNOWN_FAILURE)
- printk(KERN_DEBUG "TRANSPORT_UNKNOWN_FAILURE (%0#2x).\n",
+ printk("TRANSPORT_UNKNOWN_FAILURE (%0#2x).\n",
 req_status);
 else
- printk(KERN_DEBUG "TRANSPORT_%s.\n",
+ printk("TRANSPORT_%s.\n",
 FAIL_STATUS[req_status & 0x0F]);
```

```

/* Dump some details */
@@ -104,7 +104,7 @@ static void i2o_report_fail_status(u8 re
    printk(KERN_ERR " FailingHostUnit = 0x%04X, FailingIOP = 0x%03X\n",
           msg[5] >> 16, msg[5] & 0xFF);

- printk(KERN_ERR " Severity: 0x%02X ", (msg[4] >> 16) & 0xFF);
+ printk(KERN_ERR " Severity: 0x%02X\n", (msg[4] >> 16) & 0xFF);
  if (msg[4] & (1 << 16))
    printk(KERN_DEBUG "(FormatError), "
           "this msg can never be delivered/processed.\n");
@@ -142,9 +142,9 @@ static void i2o_report_common_status(u8
};

  if (req_status >= ARRAY_SIZE(REPLY_STATUS))
- printk(KERN_DEBUG "RequestStatus = %0#2x", req_status);
+ printk("RequestStatus = %0#2x", req_status);
  else
- printk(KERN_DEBUG "%s", REPLY_STATUS[req_status]);
+ printk("%s", REPLY_STATUS[req_status]);
}

/*
@@ -187,10 +187,10 @@ static void i2o_report_common_dsc(u16 de
};

  if (detailed_status > I2O_DSC_DEVICE_NOT_AVAILABLE)
- printk(KERN_DEBUG " / DetailedStatus = %0#4x.\n",
+ printk(" / DetailedStatus = %0#4x.\n",
        detailed_status);
  else
- printk(KERN_DEBUG " / %s.\n", COMMON_DSC[detailed_status]);
+ printk(" / %s.\n", COMMON_DSC[detailed_status]);
}

/*
@@ -200,49 +200,49 @@ static void i2o_report_util_cmd(u8 cmd)
{
  switch (cmd) {
    case I2O_CMD_UTIL_NOP:
- printk(KERN_DEBUG "UTIL_NOP, ");
+ printk("UTIL_NOP, ");
    break;
    case I2O_CMD_UTIL_ABORT:
- printk(KERN_DEBUG "UTIL_ABORT, ");
+ printk("UTIL_ABORT, ");
    break;
    case I2O_CMD_UTIL_CLAIM:

```

```

- printk(KERN_DEBUG "UTIL_CLAIM, ");
+ printk("UTIL_CLAIM, ");
  break;
  case I2O_CMD_UTIL_RELEASE:
- printk(KERN_DEBUG "UTIL_CLAIM_RELEASE, ");
+ printk("UTIL_CLAIM_RELEASE, ");
  break;
  case I2O_CMD_UTIL_CONFIG_DIALOG:
- printk(KERN_DEBUG "UTIL_CONFIG_DIALOG, ");
+ printk("UTIL_CONFIG_DIALOG, ");
  break;
  case I2O_CMD_UTIL_DEVICE_RESERVE:
- printk(KERN_DEBUG "UTIL_DEVICE_RESERVE, ");
+ printk("UTIL_DEVICE_RESERVE, ");
  break;
  case I2O_CMD_UTIL_DEVICE_RELEASE:
- printk(KERN_DEBUG "UTIL_DEVICE_RELEASE, ");
+ printk("UTIL_DEVICE_RELEASE, ");
  break;
  case I2O_CMD_UTIL_EVT_ACK:
- printk(KERN_DEBUG "UTIL_EVENT_ACKNOWLEDGE, ");
+ printk("UTIL_EVENT_ACKNOWLEDGE, ");
  break;
  case I2O_CMD_UTIL_EVT_REGISTER:
- printk(KERN_DEBUG "UTIL_EVENT_REGISTER, ");
+ printk("UTIL_EVENT_REGISTER, ");
  break;
  case I2O_CMD_UTIL_LOCK:
- printk(KERN_DEBUG "UTIL_LOCK, ");
+ printk("UTIL_LOCK, ");
  break;
  case I2O_CMD_UTIL_LOCK_RELEASE:
- printk(KERN_DEBUG "UTIL_LOCK_RELEASE, ");
+ printk("UTIL_LOCK_RELEASE, ");
  break;
  case I2O_CMD_UTIL_PARAMS_GET:
- printk(KERN_DEBUG "UTIL_PARAMS_GET, ");
+ printk("UTIL_PARAMS_GET, ");
  break;
  case I2O_CMD_UTIL_PARAMS_SET:
- printk(KERN_DEBUG "UTIL_PARAMS_SET, ");
+ printk("UTIL_PARAMS_SET, ");
  break;
  case I2O_CMD_UTIL_REPLY_FAULT_NOTIFY:
- printk(KERN_DEBUG "UTIL_REPLY_FAULT_NOTIFY, ");
+ printk("UTIL_REPLY_FAULT_NOTIFY, ");
  break;
  default:

```

```

- printk(KERN_DEBUG "Cmd = %0#2x, ", cmd);
+ printk("Cmd = %0#2x, ", cmd);
}
}

```

```

@@ -253,106 +253,106 @@ static void i2o_report_exec_cmd(u8 cmd)

```

```

{
    switch (cmd) {
        case I2O_CMD_ADAPTER_ASSIGN:
-       printk(KERN_DEBUG "EXEC_ADAPTER_ASSIGN, ");
+       printk("EXEC_ADAPTER_ASSIGN, ");
            break;
        case I2O_CMD_ADAPTER_READ:
-       printk(KERN_DEBUG "EXEC_ADAPTER_READ, ");
+       printk("EXEC_ADAPTER_READ, ");
            break;
        case I2O_CMD_ADAPTER_RELEASE:
-       printk(KERN_DEBUG "EXEC_ADAPTER_RELEASE, ");
+       printk("EXEC_ADAPTER_RELEASE, ");
            break;
        case I2O_CMD_BIOS_INFO_SET:
-       printk(KERN_DEBUG "EXEC_BIOS_INFO_SET, ");
+       printk("EXEC_BIOS_INFO_SET, ");
            break;
        case I2O_CMD_BOOT_DEVICE_SET:
-       printk(KERN_DEBUG "EXEC_BOOT_DEVICE_SET, ");
+       printk("EXEC_BOOT_DEVICE_SET, ");
            break;
        case I2O_CMD_CONFIG_VALIDATE:
-       printk(KERN_DEBUG "EXEC_CONFIG_VALIDATE, ");
+       printk("EXEC_CONFIG_VALIDATE, ");
            break;
        case I2O_CMD_CONN_SETUP:
-       printk(KERN_DEBUG "EXEC_CONN_SETUP, ");
+       printk("EXEC_CONN_SETUP, ");
            break;
        case I2O_CMD_DDM_DESTROY:
-       printk(KERN_DEBUG "EXEC_DDM_DESTROY, ");
+       printk("EXEC_DDM_DESTROY, ");
            break;
        case I2O_CMD_DDM_ENABLE:
-       printk(KERN_DEBUG "EXEC_DDM_ENABLE, ");
+       printk("EXEC_DDM_ENABLE, ");
            break;
        case I2O_CMD_DDM_QUIESCE:
-       printk(KERN_DEBUG "EXEC_DDM_QUIESCE, ");
+       printk("EXEC_DDM_QUIESCE, ");
            break;
    }
}

```

```

    case I2O_CMD_DDM_RESET:
-   printk(KERN_DEBUG "EXEC_DDM_RESET, ");
+   printk("EXEC_DDM_RESET, ");
        break;
    case I2O_CMD_DDM_SUSPEND:
-   printk(KERN_DEBUG "EXEC_DDM_SUSPEND, ");
+   printk("EXEC_DDM_SUSPEND, ");
        break;
    case I2O_CMD_DEVICE_ASSIGN:
-   printk(KERN_DEBUG "EXEC_DEVICE_ASSIGN, ");
+   printk("EXEC_DEVICE_ASSIGN, ");
        break;
    case I2O_CMD_DEVICE_RELEASE:
-   printk(KERN_DEBUG "EXEC_DEVICE_RELEASE, ");
+   printk("EXEC_DEVICE_RELEASE, ");
        break;
    case I2O_CMD_HRT_GET:
-   printk(KERN_DEBUG "EXEC_HRT_GET, ");
+   printk("EXEC_HRT_GET, ");
        break;
    case I2O_CMD_ADAPTER_CLEAR:
-   printk(KERN_DEBUG "EXEC_IOP_CLEAR, ");
+   printk("EXEC_IOP_CLEAR, ");
        break;
    case I2O_CMD_ADAPTER_CONNECT:
-   printk(KERN_DEBUG "EXEC_IOP_CONNECT, ");
+   printk("EXEC_IOP_CONNECT, ");
        break;
    case I2O_CMD_ADAPTER_RESET:
-   printk(KERN_DEBUG "EXEC_IOP_RESET, ");
+   printk("EXEC_IOP_RESET, ");
        break;
    case I2O_CMD_LCT_NOTIFY:
-   printk(KERN_DEBUG "EXEC_LCT_NOTIFY, ");
+   printk("EXEC_LCT_NOTIFY, ");
        break;
    case I2O_CMD_OUTBOUND_INIT:
-   printk(KERN_DEBUG "EXEC_OUTBOUND_INIT, ");
+   printk("EXEC_OUTBOUND_INIT, ");
        break;
    case I2O_CMD_PATH_ENABLE:
-   printk(KERN_DEBUG "EXEC_PATH_ENABLE, ");
+   printk("EXEC_PATH_ENABLE, ");
        break;
    case I2O_CMD_PATH_QUIESCE:
-   printk(KERN_DEBUG "EXEC_PATH_QUIESCE, ");
+   printk("EXEC_PATH_QUIESCE, ");
        break;

```

```

    case I2O_CMD_PATH_RESET:
-   printk(KERN_DEBUG "EXEC_PATH_RESET, ");
+   printk("EXEC_PATH_RESET, ");
        break;
    case I2O_CMD_STATIC_MF_CREATE:
-   printk(KERN_DEBUG "EXEC_STATIC_MF_CREATE, ");
+   printk("EXEC_STATIC_MF_CREATE, ");
        break;
    case I2O_CMD_STATIC_MF_RELEASE:
-   printk(KERN_DEBUG "EXEC_STATIC_MF_RELEASE, ");
+   printk("EXEC_STATIC_MF_RELEASE, ");
        break;
    case I2O_CMD_STATUS_GET:
-   printk(KERN_DEBUG "EXEC_STATUS_GET, ");
+   printk("EXEC_STATUS_GET, ");
        break;
    case I2O_CMD_SW_DOWNLOAD:
-   printk(KERN_DEBUG "EXEC_SW_DOWNLOAD, ");
+   printk("EXEC_SW_DOWNLOAD, ");
        break;
    case I2O_CMD_SW_UPLOAD:
-   printk(KERN_DEBUG "EXEC_SW_UPLOAD, ");
+   printk("EXEC_SW_UPLOAD, ");
        break;
    case I2O_CMD_SW_REMOVE:
-   printk(KERN_DEBUG "EXEC_SW_REMOVE, ");
+   printk("EXEC_SW_REMOVE, ");
        break;
    case I2O_CMD_SYS_ENABLE:
-   printk(KERN_DEBUG "EXEC_SYS_ENABLE, ");
+   printk("EXEC_SYS_ENABLE, ");
        break;
    case I2O_CMD_SYS_MODIFY:
-   printk(KERN_DEBUG "EXEC_SYS_MODIFY, ");
+   printk("EXEC_SYS_MODIFY, ");
        break;
    case I2O_CMD_SYS_QUIESCE:
-   printk(KERN_DEBUG "EXEC_SYS_QUIESCE, ");
+   printk("EXEC_SYS_QUIESCE, ");
        break;
    case I2O_CMD_SYS_TAB_SET:
-   printk(KERN_DEBUG "EXEC_SYS_TAB_SET, ");
+   printk("EXEC_SYS_TAB_SET, ");
        break;
    default:
-   printk(KERN_DEBUG "Cmd = %#02x, ", cmd);
+   printk("Cmd = %#02x, ", cmd);
}

```

```
}
```

```
@@ -361,28 +361,28 @@ void i2o_debug_state(struct i2o_controll
    printk(KERN_INFO "%s: State = ", c->name);
    switch (((i2o_status_block *) c->status_block.virt)->iop_state) {
    case 0x01:
-   printk(KERN_DEBUG "INIT\n");
+   printk("INIT\n");
        break;
    case 0x02:
-   printk(KERN_DEBUG "RESET\n");
+   printk("RESET\n");
        break;
    case 0x04:
-   printk(KERN_DEBUG "HOLD\n");
+   printk("HOLD\n");
        break;
    case 0x05:
-   printk(KERN_DEBUG "READY\n");
+   printk("READY\n");
        break;
    case 0x08:
-   printk(KERN_DEBUG "OPERATIONAL\n");
+   printk("OPERATIONAL\n");
        break;
    case 0x10:
-   printk(KERN_DEBUG "FAILED\n");
+   printk("FAILED\n");
        break;
    case 0x11:
-   printk(KERN_DEBUG "FAULTED\n");
+   printk("FAULTED\n");
        break;
    default:
-   printk(KERN_DEBUG "%x (unknown !!)\n",
+   printk("%x (unknown !!)\n",
        ((i2o_status_block *) c->status_block.virt)->iop_state);
    }
};
```
