

[patch 50/73] Input: mousedev – implement proper locking

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-02/msg02882.html>

- *From:* Greg KH <gregkh@xxxxxxx>
 - *Date:* Wed, 6 Feb 2008 15:53:44 -0800
-

2.6.23-stable review patch. If anyone has any objections, please let us know.

From: Dmitry Torokhov <dmitry.torokhov@xxxxxxxxxx>

patch 464b241575f3700e14492e34f26bcd1794280f55 in mainline.

Signed-off-by: Dmitry Torokhov <dtor@xxxxxxx>

Cc: Al Viro <viro@xxxxxxxxxxxxxxxxxxxx>

Signed-off-by: Greg Kroah-Hartman <gregkh@xxxxxxx>

drivers/input/mousedev.c | 742 ++++++-----
1 file changed, 470 insertions(+), 272 deletions(-)

```
--- a/drivers/input/mousedev.c
+++ b/drivers/input/mousedev.c
@@ -61,9 +61,11 @@ struct mousedev {
int open;
int minor;
char name[16];
+ struct input_handle handle;
wait_queue_head_t wait;
struct list_head client_list;
- struct input_handle handle;
+ spinlock_t client_lock; /* protects client_list */
+ struct mutex mutex;
struct device dev;

struct list_head mixdev_node;
@@ -113,108 +115,137 @@ static unsigned char mousedev_imex_seq[]
static struct input_handler mousedev_handler;

static struct mousedev *mousedev_table[MOUSEDEV_MINORS];
+static DEFINE_MUTEX(mousedev_table_mutex);
static struct mousedev *mousedev_mix;
static LIST_HEAD(mousedev_mix_list);

+static void mixdev_open_devices(void);
```

[patch 50/73] Input: mousedev – implement proper locking

```
+static void mixdev_close_devices(void);
+
#define fx(i) (mousedev->old_x[(mousedev->pkt_count - (i)) & 03])
#define fy(i) (mousedev->old_y[(mousedev->pkt_count - (i)) & 03])

-static void mousedev_touchpad_event(struct input_dev *dev, struct mousedev *mousedev, unsigned int
code, int value)
+static void mousedev_touchpad_event(struct input_dev *dev,
+ struct mousedev *mousedev,
+ unsigned int code, int value)
{
int size, tmp;
enum { FRACTION_DENOM = 128 };

switch (code) {
- case ABS_X:
- fx(0) = value;
- if (mousedev->touch && mousedev->pkt_count >= 2) {
- size = dev->absmax[ABS_X] - dev->absmin[ABS_X];
- if (size == 0)
- size = 256 * 2;
- tmp = ((value - fx(2)) * (256 * FRACTION_DENOM)) / size;
- tmp += mousedev->frac_dx;
- mousedev->packet.dx = tmp / FRACTION_DENOM;
- mousedev->frac_dx = tmp - mousedev->packet.dx * FRACTION_DENOM;
- }
- break;

- case ABS_Y:
- fy(0) = value;
- if (mousedev->touch && mousedev->pkt_count >= 2) {
- /* use X size to keep the same scale */
- size = dev->absmax[ABS_X] - dev->absmin[ABS_X];
- if (size == 0)
- size = 256 * 2;
- tmp = -((value - fy(2)) * (256 * FRACTION_DENOM)) / size;
- tmp += mousedev->frac_dy;
- mousedev->packet.dy = tmp / FRACTION_DENOM;
- mousedev->frac_dy = tmp - mousedev->packet.dy * FRACTION_DENOM;
- }
- break;
+ case ABS_X:
+ fx(0) = value;
+ if (mousedev->touch && mousedev->pkt_count >= 2) {
+ size = dev->absmax[ABS_X] - dev->absmin[ABS_X];
+ if (size == 0)
+ size = 256 * 2;
+ tmp = ((value - fx(2)) * 256 * FRACTION_DENOM) / size;
+ tmp += mousedev->frac_dx;
+ mousedev->packet.dx = tmp / FRACTION_DENOM;
+ mousedev->frac_dx =
```

[patch 50/73] Input: mousedev – implement proper locking

```
+ tmp = mousedev->packet.dx * FRACTION_DENOM;
+ }
+ break;
+
+ case ABS_Y:
+ fy(0) = value;
+ if (mousedev->touch && mousedev->pkt_count >= 2) {
+ /* use X size to keep the same scale */
+ size = dev->absmax[ABS_X] - dev->absmin[ABS_X];
+ if (size == 0)
+ size = 256 * 2;
+ tmp = -((value - fy(2)) * 256 * FRACTION_DENOM) / size;
+ tmp += mousedev->frac_dy;
+ mousedev->packet.dy = tmp / FRACTION_DENOM;
+ mousedev->frac_dy = tmp -
+ mousedev->packet.dy * FRACTION_DENOM;
+ }
+ break;
+ }
+ }
```

```
-static void mousedev_abs_event(struct input_dev *dev, struct mousedev *mousedev, unsigned int code, int value)
```

```
+static void mousedev_abs_event(struct input_dev *dev, struct mousedev *mousedev,
+ unsigned int code, int value)
```

```
{
int size;
```

```
switch (code) {
```

```
- case ABS_X:
```

```
- size = dev->absmax[ABS_X] - dev->absmin[ABS_X];
```

```
- if (size == 0)
```

```
- size = xres ? : 1;
```

```
- if (value > dev->absmax[ABS_X])
```

```
- value = dev->absmax[ABS_X];
```

```
- if (value < dev->absmin[ABS_X])
```

```
- value = dev->absmin[ABS_X];
```

```
- mousedev->packet.x = ((value - dev->absmin[ABS_X]) * xres) / size;
```

```
- mousedev->packet.abs_event = 1;
```

```
- break;
```

```
- case ABS_Y:
```

```
- size = dev->absmax[ABS_Y] - dev->absmin[ABS_Y];
```

```
- if (size == 0)
```

```
- size = yres ? : 1;
```

```
- if (value > dev->absmax[ABS_Y])
```

```
- value = dev->absmax[ABS_Y];
```

```
- if (value < dev->absmin[ABS_Y])
```

```
- value = dev->absmin[ABS_Y];
```

```
- mousedev->packet.y = yres - ((value - dev->absmin[ABS_Y]) * yres) / size;
```

```
- mousedev->packet.abs_event = 1;
```

[patch 50/73] Input: mousedev – implement proper locking

```
- break;
+ case ABS_X:
+ size = dev->absmax[ABS_X] - dev->absmin[ABS_X];
+ if (size == 0)
+ size = xres ? : 1;
+ if (value > dev->absmax[ABS_X])
+ value = dev->absmax[ABS_X];
+ if (value < dev->absmin[ABS_X])
+ value = dev->absmin[ABS_X];
+ mousedev->packet.x =
+ ((value - dev->absmin[ABS_X]) * xres) / size;
+ mousedev->packet.abs_event = 1;
+ break;
+
+ case ABS_Y:
+ size = dev->absmax[ABS_Y] - dev->absmin[ABS_Y];
+ if (size == 0)
+ size = yres ? : 1;
+ if (value > dev->absmax[ABS_Y])
+ value = dev->absmax[ABS_Y];
+ if (value < dev->absmin[ABS_Y])
+ value = dev->absmin[ABS_Y];
+ mousedev->packet.y = yres -
+ ((value - dev->absmin[ABS_Y]) * yres) / size;
+ mousedev->packet.abs_event = 1;
+ break;
}
}

-static void mousedev_rel_event(struct mousedev *mousedev, unsigned int code, int value)
+static void mousedev_rel_event(struct mousedev *mousedev,
+ unsigned int code, int value)
{
switch (code) {
- case REL_X: mousedev->packet.dx += value; break;
- case REL_Y: mousedev->packet.dy -= value; break;
- case REL_WHEEL: mousedev->packet.dz -= value; break;
+ case REL_X:
+ mousedev->packet.dx += value;
+ break;
+
+ case REL_Y:
+ mousedev->packet.dy -= value;
+ break;
+
+ case REL_WHEEL:
+ mousedev->packet.dz -= value;
+ break;
}
}
```

[patch 50/73] Input: mousedev – implement proper locking

```
-static void mousedev_key_event(struct mousedev *mousedev, unsigned int code, int value)
+static void mousedev_key_event(struct mousedev *mousedev,
+ unsigned int code, int value)
{
int index;

switch (code) {
- case BTN_TOUCH:
- case BTN_0:
- case BTN_LEFT: index = 0; break;
- case BTN_STYLUS:
- case BTN_1:
- case BTN_RIGHT: index = 1; break;
- case BTN_2:
- case BTN_FORWARD:
- case BTN_STYLUS2:
- case BTN_MIDDLE: index = 2; break;
- case BTN_3:
- case BTN_BACK:
- case BTN_SIDE: index = 3; break;
- case BTN_4:
- case BTN_EXTRA: index = 4; break;
- default: return;
+
+ case BTN_TOUCH:
+ case BTN_0:
+ case BTN_LEFT: index = 0; break;
+
+ case BTN_STYLUS:
+ case BTN_1:
+ case BTN_RIGHT: index = 1; break;
+
+ case BTN_2:
+ case BTN_FORWARD:
+ case BTN_STYLUS2:
+ case BTN_MIDDLE: index = 2; break;
+
+ case BTN_3:
+ case BTN_BACK:
+ case BTN_SIDE: index = 3; break;
+
+ case BTN_4:
+ case BTN_EXTRA: index = 4; break;
+
+ default: return;
}

if (value) {
@@ -226,19 +257,22 @@ static void mousedev_key_event(struct mo
}
}
```

[patch 50/73] Input: mousedev – implement proper locking

```
–static void mousedev_notify_readers(struct mousedev *mousedev, struct mousedev_hw_data *packet)
+static void mousedev_notify_readers(struct mousedev *mousedev,
+ struct mousedev_hw_data *packet)
{
struct mousedev_client *client;
struct mousedev_motion *p;
– unsigned long flags;
+ unsigned int new_head;
int wake_readers = 0;

– list_for_each_entry(client, &mousedev->client_list, node) {
– spin_lock_irqsave(&client->packet_lock, flags);
+ list_for_each_entry_rcu(client, &mousedev->client_list, node) {
+
+ /* Just acquire the lock, interrupts already disabled */
+ spin_lock(&client->packet_lock);

p = &client->packets[client->head];
if (client->ready && p->buttons != mousedev->packet.buttons) {
– unsigned int new_head = (client->head + 1) % PACKET_QUEUE_LEN;
+ new_head = (client->head + 1) % PACKET_QUEUE_LEN;
if (new_head != client->tail) {
p = &client->packets[client->head = new_head];
memset(p, 0, sizeof(struct mousedev_motion));
@@ -253,19 +287,22 @@ static void mousedev_notify_readers(stru
}

client->pos_x += packet->dx;
– client->pos_x = client->pos_x < 0 ? 0 : (client->pos_x >= xres ? xres : client->pos_x);
+ client->pos_x = client->pos_x < 0 ?
+ 0 : (client->pos_x >= xres ? xres : client->pos_x);
client->pos_y += packet->dy;
– client->pos_y = client->pos_y < 0 ? 0 : (client->pos_y >= yres ? yres : client->pos_y);
+ client->pos_y = client->pos_y < 0 ?
+ 0 : (client->pos_y >= yres ? yres : client->pos_y);

p->dx += packet->dx;
p->dy += packet->dy;
p->dz += packet->dz;
p->buttons = mousedev->packet.buttons;

– if (p->dx || p->dy || p->dz || p->buttons != client->last_buttons)
+ if (p->dx || p->dy || p->dz ||
+ p->buttons != client->last_buttons)
client->ready = 1;

– spin_unlock_irqrestore(&client->packet_lock, flags);
+ spin_unlock(&client->packet_lock);

if (client->ready) {
```

[patch 50/73] Input: mousedev – implement proper locking

```
kill_fasync(&client->fasync, SIGIO, POLL_IN);
@@ -281,7 +318,8 @@ static void mousedev_touchpad_touch(stru
{
if (!value) {
if (mousedev->touch &&
- time_before(jiffies, mousedev->touch + msecs_to_jiffies(tap_time))) {
+ time_before(jiffies,
+ mousedev->touch + msecs_to_jiffies(tap_time))) {
/*
* Toggle left button to emulate tap.
* We rely on the fact that mousedev_mix always has 0
@@ -290,7 +328,8 @@ static void mousedev_touchpad_touch(stru
set_bit(0, &mousedev->packet.buttons);
set_bit(0, &mousedev_mix->packet.buttons);
mousedev_notify_readers(mousedev, &mousedev_mix->packet);
- mousedev_notify_readers(mousedev_mix, &mousedev_mix->packet);
+ mousedev_notify_readers(mousedev_mix,
+ &mousedev_mix->packet);
clear_bit(0, &mousedev->packet.buttons);
clear_bit(0, &mousedev_mix->packet.buttons);
}
@@ -302,54 +341,61 @@ static void mousedev_touchpad_touch(stru
mousedev->touch = jiffies;
}

-static void mousedev_event(struct input_handle *handle, unsigned int type, unsigned int code, int value)
+static void mousedev_event(struct input_handle *handle,
+ unsigned int type, unsigned int code, int value)
{
struct mousedev *mousedev = handle->private;

switch (type) {
- case EV_ABS:
- /* Ignore joysticks */
- if (test_bit(BTN_TRIGGER, handle->dev->keybit))
- return;

- if (test_bit(BTN_TOOL_FINGER, handle->dev->keybit))
- mousedev_touchpad_event(handle->dev, mousedev, code, value);
+ case EV_ABS:
+ /* Ignore joysticks */
+ if (test_bit(BTN_TRIGGER, handle->dev->keybit))
+ return;
+
+ if (test_bit(BTN_TOOL_FINGER, handle->dev->keybit))
+ mousedev_touchpad_event(handle->dev,
+ mousedev, code, value);
+ else
+ mousedev_abs_event(handle->dev, mousedev, code, value);
+
+ break;
}
```

[patch 50/73] Input: mousedev – implement proper locking

```
+
+ case EV_REL:
+ mousedev_rel_event(mousedev, code, value);
+ break;
+
+ case EV_KEY:
+ if (value != 2) {
+ if (code == BTN_TOUCH &&
+ test_bit(BTN_TOOL_FINGER, handle->dev->keybit))
+ mousedev_touchpad_touch(mousedev, value);
else
- mousedev_abs_event(handle->dev, mousedev, code, value);
-
- break;
-
- case EV_REL:
- mousedev_rel_event(mousedev, code, value);
- break;
+ mousedev_key_event(mousedev, code, value);
+ }
+ break;

- case EV_KEY:
- if (value != 2) {
- if (code == BTN_TOUCH && test_bit(BTN_TOOL_FINGER, handle->dev->keybit))
- mousedev_touchpad_touch(mousedev, value);
- else
- mousedev_key_event(mousedev, code, value);
+ case EV_SYN:
+ if (code == SYN_REPORT) {
+ if (mousedev->touch) {
+ mousedev->pkt_count++;
+ /*
+ * Input system eats duplicate events,
+ * but we need all of them to do correct
+ * averaging so apply present one forward
+ */
+ fx(0) = fx(1);
+ fy(0) = fy(1);
+ }
- break;
-
- case EV_SYN:
- if (code == SYN_REPORT) {
- if (mousedev->touch) {
- mousedev->pkt_count++;
- /* Input system eats duplicate events, but we need all of them
- * to do correct averaging so apply present one forward
- */
- fx(0) = fx(1);
- fy(0) = fy(1);
```

[patch 50/73] Input: mousedev – implement proper locking

```
- }

- mousedev_notify_readers(mousedev, &mousedev->packet);
- mousedev_notify_readers(mousedev_mix, &mousedev->packet);
+ mousedev_notify_readers(mousedev, &mousedev->packet);
+ mousedev_notify_readers(mousedev_mix, &mousedev->packet);

- mousedev->packet.dx = mousedev->packet.dy = mousedev->packet.dz = 0;
- mousedev->packet.abs_event = 0;
- }
- break;
+ mousedev->packet.dx = mousedev->packet.dy =
+ mousedev->packet.dz = 0;
+ mousedev->packet.abs_event = 0;
+ }
+ break;
}
}

@@ -367,41 +413,45 @@ static void mousedev_free(struct device
{
struct mousedev *mousedev = container_of(dev, struct mousedev, dev);

- mousedev_table[mousedev->minor] = NULL;
kfree(mousedev);
}

-static int mixdev_add_device(struct mousedev *mousedev)
+static int mousedev_open_device(struct mousedev *mousedev)
{
- int error;
-
- if (mousedev_mix->open) {
- error = input_open_device(&mousedev->handle);
- if (error)
- return error;
+ int retval;

- mousedev->open++;
- mousedev->mixdev_open = 1;
- }
+ retval = mutex_lock_interruptible(&mousedev->mutex);
+ if (retval)
+ return retval;

- get_device(&mousedev->dev);
- list_add_tail(&mousedev->mixdev_node, &mousedev_mix_list);
+ if (mousedev->minor == MOUSEDEV_MIX)
+ mixdev_open_devices();
+ else if (!mousedev->exist)
+ retval = -ENODEV;
```

[patch 50/73] Input: mousedev – implement proper locking

```
+ else if (!mousedev->open++)
+ retval = input_open_device(&mousedev->handle);

- return 0;
+ mutex_unlock(&mousedev->mutex);
+ return retval;
}

-static void mixdev_remove_device(struct mousedev *mousedev)
+static void mousedev_close_device(struct mousedev *mousedev)
{
- if (mousedev->mixdev_open) {
- mousedev->mixdev_open = 0;
- if (!--mousedev->open && mousedev->exist)
- input_close_device(&mousedev->handle);
- }
+ mutex_lock(&mousedev->mutex);

- list_del_init(&mousedev->mixdev_node);
- put_device(&mousedev->dev);
+ if (mousedev->minor == MOUSEDEV_MIX)
+ mixdev_close_devices();
+ else if (mousedev->exist && !--mousedev->open)
+ input_close_device(&mousedev->handle);
+
+ mutex_unlock(&mousedev->mutex);
}

+/*
+ * Open all available devices so they can all be multiplexed in one.
+ * stream. Note that this function is called with mousedev_mix->mutex
+ * held.
+ */
static void mixdev_open_devices(void)
{
struct mousedev *mousedev;
@@ -411,16 +461,19 @@ static void mixdev_open_devices(void)

list_for_each_entry(mousedev, &mousedev_mix_list, mixdev_node) {
if (!mousedev->mixdev_open) {
- if (!mousedev->open && mousedev->exist)
- if (input_open_device(&mousedev->handle))
- continue;
+ if (mousedev_open_device(mousedev))
+ continue;

- mousedev->open++;
mousedev->mixdev_open = 1;
}
}
}
```

[patch 50/73] Input: mousedev – implement proper locking

```
+/*
+ * Close all devices that were opened as part of multiplexed
+ * device. Note that this function is called with mousedev_mix->mutex
+ * held.
+ */
static void mixdev_close_devices(void)
{
struct mousedev *mousedev;
@@ -431,33 +484,50 @@ static void mixdev_close_devices(void)
list_for_each_entry(mousedev, &mousedev_mix_list, mixdev_node) {
if (mousedev->mixdev_open) {
mousedev->mixdev_open = 0;
- if (!--mousedev->open && mousedev->exist)
- input_close_device(&mousedev->handle);
+ mousedev_close_device(mousedev);
}
}
}

+
+static void mousedev_attach_client(struct mousedev *mousedev,
+ struct mousedev_client *client)
+{
+ spin_lock(&mousedev->client_lock);
+ list_add_tail_rcu(&client->node, &mousedev->client_list);
+ spin_unlock(&mousedev->client_lock);
+ /*
+ * We don't use synchronize_rcu() here because read-side
+ * critical section is protected by a spinlock (dev->event_lock)
+ * instead of rcu_read_lock().
+ */
+ synchronize_sched();
+}
+
+static void mousedev_detach_client(struct mousedev *mousedev,
+ struct mousedev_client *client)
+{
+ spin_lock(&mousedev->client_lock);
+ list_del_rcu(&client->node);
+ spin_unlock(&mousedev->client_lock);
+ synchronize_sched();
+}
+
static int mousedev_release(struct inode *inode, struct file *file)
{
struct mousedev_client *client = file->private_data;
struct mousedev *mousedev = client->mousedev;

mousedev_fasync(-1, file, 0);
-
```

[patch 50/73] Input: mousetdev – implement proper locking

```
- list_del(&client->node);
+ mousetdev_detach_client(mousetdev, client);
kfree(client);

- if (mousetdev->minor == MOUSEDEV_MIX)
- mixdev_close_devices();
- else if (!--mousetdev->open && mousetdev->exist)
- input_close_device(&mousetdev->handle);
-
+ mousetdev_close_device(mousetdev);
put_device(&mousetdev->dev);

return 0;
}

-
static int mousetdev_open(struct inode *inode, struct file *file)
{
struct mousetdev_client *client;
@@ -475,12 +545,17 @@ static int mousetdev_open(struct inode *i
if (i >= MOUSEDEV_MINORS)
return -ENODEV;

+ error = mutex_lock_interruptible(&mousetdev_table_mutex);
+ if (error)
+ return error;
mousetdev = mousetdev_table[i];
+ if (mousetdev)
+ get_device(&mousetdev->dev);
+ mutex_unlock(&mousetdev_table_mutex);
+
if (!mousetdev)
return -ENODEV;

- get_device(&mousetdev->dev);
-
client = kzalloc(sizeof(struct mousetdev_client), GFP_KERNEL);
if (!client) {
error = -ENOMEM;
@@ -491,21 +566,17 @@ static int mousetdev_open(struct inode *i
client->pos_x = xres / 2;
client->pos_y = yres / 2;
client->mousetdev = mousetdev;
- list_add_tail(&client->node, &mousetdev->client_list);
+ mousetdev_attach_client(mousetdev, client);

- if (mousetdev->minor == MOUSEDEV_MIX)
- mixdev_open_devices();
- else if (!mousetdev->open++ && mousetdev->exist) {
- error = input_open_device(&mousetdev->handle);
- if (error)
```

[patch 50/73] Input: mousedev – implement proper locking

```
- goto err_free_client;
- }
+ error = mousedev_open_device(mousedev);
+ if (error)
+ goto err_free_client;

file->private_data = client;
return 0;

err_free_client:
- list_del(&client->node);
+ mousedev_detach_client(mousedev, client);
kfree(client);
err_put_mousedev:
put_device(&mousedev->dev);
@@ -517,41 +588,41 @@ static inline int mousedev_limit_delta(i
return delta > limit ? limit : (delta < -limit ? -limit : delta);
}

-static void mousedev_packet(struct mousedev_client *client, signed char *ps2_data)
+static void mousedev_packet(struct mousedev_client *client,
+ signed char *ps2_data)
{
- struct mousedev_motion *p;
- unsigned long flags;
-
- spin_lock_irqsave(&client->packet_lock, flags);
- p = &client->packets[client->tail];
+ struct mousedev_motion *p = &client->packets[client->tail];

- ps2_data[0] = 0x08 | ((p->dx < 0) << 4) | ((p->dy < 0) << 5) | (p->buttons & 0x07);
+ ps2_data[0] = 0x08 |
+ ((p->dx < 0) << 4) | ((p->dy < 0) << 5) | (p->buttons & 0x07);
ps2_data[1] = mousedev_limit_delta(p->dx, 127);
ps2_data[2] = mousedev_limit_delta(p->dy, 127);
p->dx -= ps2_data[1];
p->dy -= ps2_data[2];

switch (client->mode) {
- case MOUSEDEV_EMUL_EXPS:
- ps2_data[3] = mousedev_limit_delta(p->dz, 7);
- p->dz -= ps2_data[3];
- ps2_data[3] = (ps2_data[3] & 0x0f) | ((p->buttons & 0x18) << 1);
- client->bufsiz = 4;
- break;
-
- case MOUSEDEV_EMUL_IMPS:
- ps2_data[0] |= ((p->buttons & 0x10) >> 3) | ((p->buttons & 0x08) >> 1);
- ps2_data[3] = mousedev_limit_delta(p->dz, 127);
- p->dz -= ps2_data[3];
- client->bufsiz = 4;
```

[patch 50/73] Input: mousedev – implement proper locking

```
- break;
-
- case MOUSEDEV_EMUL_PS2:
- default:
- ps2_data[0] |= ((p->buttons & 0x10) >> 3) | ((p->buttons & 0x08) >> 1);
- p->dz = 0;
- client->bufsiz = 3;
- break;
+ case MOUSEDEV_EMUL_EXPS:
+ ps2_data[3] = mousedev_limit_delta(p->dz, 7);
+ p->dz -= ps2_data[3];
+ ps2_data[3] = (ps2_data[3] & 0x0f) | ((p->buttons & 0x18) << 1);
+ client->bufsiz = 4;
+ break;
+
+ case MOUSEDEV_EMUL_IMPS:
+ ps2_data[0] |=
+ ((p->buttons & 0x10) >> 3) | ((p->buttons & 0x08) >> 1);
+ ps2_data[3] = mousedev_limit_delta(p->dz, 127);
+ p->dz -= ps2_data[3];
+ client->bufsiz = 4;
+ break;
+
+ case MOUSEDEV_EMUL_PS2:
+ default:
+ ps2_data[0] |=
+ ((p->buttons & 0x10) >> 3) | ((p->buttons & 0x08) >> 1);
+ p->dz = 0;
+ client->bufsiz = 3;
+ break;
}

if (!p->dx && !p->dy && !p->dz) {
@@ -561,12 +632,56 @@ static void mousedev_packet(struct mouse
} else
client->tail = (client->tail + 1) % PACKET_QUEUE_LEN;
}
-
- spin_unlock_irqrestore(&client->packet_lock, flags);
}

+static void mousedev_generate_response(struct mousedev_client *client,
+ int command)
+{
+ client->ps2[0] = 0xfa; /* ACK */
+
+ switch (command) {

-static ssize_t mousedev_write(struct file *file, const char __user *buffer, size_t count, loff_t *ppos)
+ case 0xeb: /* Poll */
+ mousedev_packet(client, &client->ps2[1]);
```

[patch 50/73] Input: mousedev – implement proper locking

```
+ client->bufsiz++; /* account for leading ACK */
+ break;
+
+ case 0xf2: /* Get ID */
+ switch (client->mode) {
+ case MOUSEDEV_EMUL_PS2:
+ client->ps2[1] = 0;
+ break;
+ case MOUSEDEV_EMUL_IMPS:
+ client->ps2[1] = 3;
+ break;
+ case MOUSEDEV_EMUL_EXPS:
+ client->ps2[1] = 4;
+ break;
+ }
+ client->bufsiz = 2;
+ break;
+
+ case 0xe9: /* Get info */
+ client->ps2[1] = 0x60; client->ps2[2] = 3; client->ps2[3] = 200;
+ client->bufsiz = 4;
+ break;
+
+ case 0xff: /* Reset */
+ client->impsseq = client->imexseq = 0;
+ client->mode = MOUSEDEV_EMUL_PS2;
+ client->ps2[1] = 0xaa; client->ps2[2] = 0x00;
+ client->bufsiz = 3;
+ break;
+
+ default:
+ client->bufsiz = 1;
+ break;
+ }
+ client->buffer = client->bufsiz;
+}
+
+static ssize_t mousedev_write(struct file *file, const char __user *buffer,
+ size_t count, loff_t *ppos)
+{
+ struct mousedev_client *client = file->private_data;
+ unsigned char c;
+@@ -577,6 +692,8 @@ static ssize_t mousedev_write(struct fil
+ if (get_user(c, buffer + i))
+ return -EFAULT;
+
+ spin_lock_irq(&client->packet_lock);
+
+ if (c == mousedev_imex_seq[client->imexseq]) {
+ if (++client->imexseq == MOUSEDEV_SEQ_LEN) {
+ client->imexseq = 0;
```

[patch 50/73] Input: mousedev – implement proper locking

```
@@ -593,68 +710,39 @@ static ssize_t mousedev_write(struct fil
} else
client->impsseq = 0;

- client->ps2[0] = 0xfa;
-
- switch (c) {
-
- case 0xeb: /* Poll */
- mousedev_packet(client, &client->ps2[1]);
- client->bufsiz++; /* account for leading ACK */
- break;
-
- case 0xf2: /* Get ID */
- switch (client->mode) {
- case MOUSEDEV_EMUL_PS2: client->ps2[1] = 0; break;
- case MOUSEDEV_EMUL_IMPS: client->ps2[1] = 3; break;
- case MOUSEDEV_EMUL_EXPS: client->ps2[1] = 4; break;
- }
- client->bufsiz = 2;
- break;
-
- case 0xe9: /* Get info */
- client->ps2[1] = 0x60; client->ps2[2] = 3; client->ps2[3] = 200;
- client->bufsiz = 4;
- break;
-
- case 0xff: /* Reset */
- client->impsseq = client->imexseq = 0;
- client->mode = MOUSEDEV_EMUL_PS2;
- client->ps2[1] = 0xaa; client->ps2[2] = 0x00;
- client->bufsiz = 3;
- break;
-
- default:
- client->bufsiz = 1;
- break;
- }
+ mousedev_generate_response(client, c);

- client->buffer = client->bufsiz;
+ spin_unlock_irq(&client->packet_lock);
}

kill_fasync(&client->fasync, SIGIO, POLL_IN);
-
wake_up_interruptible(&client->mousedev->wait);

return count;
}
```

[patch 50/73] Input: mousedev – implement proper locking

```
-static ssize_t mousedev_read(struct file *file, char __user *buffer, size_t count, loff_t *ppos)
+static ssize_t mousedev_read(struct file *file, char __user *buffer,
+ size_t count, loff_t *ppos)
{
struct mousedev_client *client = file->private_data;
+ struct mousedev *mousedev = client->mousedev;
+ signed char data[sizeof(client->ps2)];
int retval = 0;

- if (!client->ready && !client->buffer && (file->f_flags & O_NONBLOCK))
+ if (!client->ready && !client->buffer && mousedev->exist &&
+ (file->f_flags & O_NONBLOCK))
return -EAGAIN;

- retval = wait_event_interruptible(client->mousedev->wait,
- !client->mousedev->exist || client->ready || client->buffer);
-
+ retval = wait_event_interruptible(mousedev->wait,
+ !mousedev->exist || client->ready || client->buffer);
if (retval)
return retval;

- if (!client->mousedev->exist)
+ if (!mousedev->exist)
return -ENODEV;

+ spin_lock_irq(&client->packet_lock);
+
if (!client->buffer && client->ready) {
mousedev_packet(client, client->ps2);
client->buffer = client->bufsiz;
@@ -663,9 +751,12 @@ static ssize_t mousedev_read(struct file
if (count > client->buffer)
count = client->buffer;

+ memcpy(data, client->ps2 + client->bufsiz - client->buffer, count);
client->buffer -= count;

- if (copy_to_user(buffer, client->ps2 + client->bufsiz - client->buffer - count, count))
+ spin_unlock_irq(&client->packet_lock);
+
+ if (copy_to_user(buffer, data, count))
return -EFAULT;

return count;
@@ -692,6 +783,60 @@ static const struct file_operations mous
.fasync = mousedev_fasync,
};

+static int mousedev_install_chrdev(struct mousedev *mousedev)
+{
```

[patch 50/73] Input: mousedev – implement proper locking

```
+ mousedev_table[mousedev->minor] = mousedev;
+ return 0;
+}
+
+static void mousedev_remove_chrdev(struct mousedev *mousedev)
+{
+ mutex_lock(&mousedev_table_mutex);
+ mousedev_table[mousedev->minor] = NULL;
+ mutex_unlock(&mousedev_table_mutex);
+}
+
+/*
+ * Mark device non-existent. This disables writes, ioctls and
+ * prevents new users from opening the device. Already posted
+ * blocking reads will stay, however new ones will fail.
+ */
+static void mousedev_mark_dead(struct mousedev *mousedev)
+{
+ mutex_lock(&mousedev->mutex);
+ mousedev->exist = 0;
+ mutex_unlock(&mousedev->mutex);
+}
+
+/*
+ * Wake up users waiting for IO so they can disconnect from
+ * dead device.
+ */
+static void mousedev_hangup(struct mousedev *mousedev)
+{
+ struct mousedev_client *client;
+
+ spin_lock(&mousedev->client_lock);
+ list_for_each_entry(client, &mousedev->client_list, node)
+ kill_fasync(&client->fasync, SIGIO, POLL_HUP);
+ spin_unlock(&mousedev->client_lock);
+
+ wake_up_interruptible(&mousedev->wait);
+}
+
+static void mousedev_cleanup(struct mousedev *mousedev)
+{
+ struct input_handle *handle = &mousedev->handle;
+
+ mousedev_mark_dead(mousedev);
+ mousedev_hangup(mousedev);
+ mousedev_remove_chrdev(mousedev);
+
+ /* mousedev is marked dead so no one else accesses mousedev->open */
+ if (mousedev->open)
+ input_close_device(handle);
+}
+}
```

[patch 50/73] Input: mousedev – implement proper locking

```
+
static struct mousedev *mousedev_create(struct input_dev *dev,
struct input_handler *handler,
int minor)
@@ -707,6 +852,10 @@ static struct mousedev *mousedev_create(

INIT_LIST_HEAD(&mousedev->client_list);
INIT_LIST_HEAD(&mousedev->mixdev_node);
+ spin_lock_init(&mousedev->client_lock);
+ mutex_init(&mousedev->mutex);
+ lockdep_set_subclass(&mousedev->mutex,
+ minor == MOUSEDEV_MIX ? MOUSEDEV_MIX : 0);
init_waitqueue_head(&mousedev->wait);

if (minor == MOUSEDEV_MIX)
@@ -731,14 +880,27 @@ static struct mousedev *mousedev_create(
mousedev->dev.release = mousedev_free;
device_initialize(&mousedev->dev);

- mousedev_table[minor] = mousedev;
+ if (minor != MOUSEDEV_MIX) {
+ error = input_register_handle(&mousedev->handle);
+ if (error)
+ goto err_free_mousedev;
+ }
+
+ error = mousedev_install_chrdev(mousedev);
+ if (error)
+ goto err_unregister_handle;

error = device_add(&mousedev->dev);
if (error)
- goto err_free_mousedev;
+ goto err_cleanup_mousedev;

return mousedev;

+ err_cleanup_mousedev:
+ mousedev_cleanup(mousedev);
+ err_unregister_handle:
+ if (minor != MOUSEDEV_MIX)
+ input_unregister_handle(&mousedev->handle);
err_free_mousedev:
put_device(&mousedev->dev);
err_out:
@@ -747,29 +909,64 @@ static struct mousedev *mousedev_create(

static void mousedev_destroy(struct mousedev *mousedev)
{
- struct mousedev_client *client;
-

```

[patch 50/73] Input: mousedev – implement proper locking

```
device_del(&mousedev->dev);
- mousedev->exist = 0;
+ mousedev_cleanup(mousedev);
+ if (mousedev->minor != MOUSEDEV_MIX)
+ input_unregister_handle(&mousedev->handle);
+ put_device(&mousedev->dev);
+}

- if (mousedev->open) {
- input_close_device(&mousedev->handle);
- list_for_each_entry(client, &mousedev->client_list, node)
- kill_fasync(&client->fasync, SIGIO, POLL_HUP);
- wake_up_interruptible(&mousedev->wait);
+static int mixdev_add_device(struct mousedev *mousedev)
+{
+ int retval;
+
+ retval = mutex_lock_interruptible(&mousedev_mix->mutex);
+ if (retval)
+ return retval;
+
+ if (mousedev_mix->open) {
+ retval = mousedev_open_device(mousedev);
+ if (retval)
+ goto out;
+
+ mousedev->mixdev_open = 1;
+ }

+ get_device(&mousedev->dev);
+ list_add_tail(&mousedev->mixdev_node, &mousedev_mix_list);
+
+ out:
+ mutex_unlock(&mousedev_mix->mutex);
+ return retval;
+}

+static void mixdev_remove_device(struct mousedev *mousedev)
+{
+ mutex_lock(&mousedev_mix->mutex);
+
+ if (mousedev->mixdev_open) {
+ mousedev->mixdev_open = 0;
+ mousedev_close_device(mousedev);
+ }
+
+ list_del_init(&mousedev->mixdev_node);
+ mutex_unlock(&mousedev_mix->mutex);
+
+ put_device(&mousedev->dev);
+}

```

[patch 50/73] Input: mousedev – implement proper locking

```
-static int mousedev_connect(struct input_handler *handler, struct input_dev *dev,
+static int mousedev_connect(struct input_handler *handler,
+ struct input_dev *dev,
const struct input_device_id *id)
{
struct mousedev *mousedev;
int minor;
int error;

- for (minor = 0; minor < MOUSEDEV_MINORS && mousedev_table[minor]; minor++);
+ for (minor = 0; minor < MOUSEDEV_MINORS; minor++)
+ if (!mousedev_table[minor])
+ break;
+
if (minor == MOUSEDEV_MINORS) {
printk(KERN_ERR "mousedev: no more free mousedev devices\n");
return -ENFILE;
@@ -779,21 +976,13 @@ static int mousedev_connect(struct input
if (IS_ERR(mousedev))
return PTR_ERR(mousedev);

- error = input_register_handle(&mousedev->handle);
- if (error)
- goto err_delete_mousedev;
-
error = mixdev_add_device(mousedev);
- if (error)
- goto err_unregister_handle;
+ if (error) {
+ mousedev_destroy(mousedev);
+ return error;
+ }

return 0;
-
- err_unregister_handle:
- input_unregister_handle(&mousedev->handle);
- err_delete_mousedev:
- device_unregister(&mousedev->dev);
- return error;
}

static void mousedev_disconnect(struct input_handle *handle)
@@ -801,33 +990,42 @@ static void mousedev_disconnect(struct i
struct mousedev *mousedev = handle->private;

mixdev_remove_device(mousedev);
- input_unregister_handle(handle);
mousedev_destroy(mousedev);
}
```

[patch 50/73] Input: mousedev – implement proper locking

```
static const struct input_device_id mousedev_ids[] = {
{
- .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT |
INPUT_DEVICE_ID_MATCH_RELBIT,
+ .flags = INPUT_DEVICE_ID_MATCH_EVBIT |
+ INPUT_DEVICE_ID_MATCH_KEYBIT |
+ INPUT_DEVICE_ID_MATCH_RELBIT,
.evbit = { BIT(EV_KEY) | BIT(EV_REL) },
.keybit = { [LONG(BTN_LEFT)] = BIT(BTN_LEFT) },
.relbit = { BIT(REL_X) | BIT(REL_Y) },
- }, /* A mouse like device, at least one button, two relative axes */
+ }, /* A mouse like device, at least one button,
+ two relative axes */
{
- .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_RELBIT,
+ .flags = INPUT_DEVICE_ID_MATCH_EVBIT |
+ INPUT_DEVICE_ID_MATCH_RELBIT,
.evbit = { BIT(EV_KEY) | BIT(EV_REL) },
.relbit = { BIT(REL_WHEEL) },
}, /* A separate scrollwheel */
{
- .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT |
INPUT_DEVICE_ID_MATCH_ABSBIT,
+ .flags = INPUT_DEVICE_ID_MATCH_EVBIT |
+ INPUT_DEVICE_ID_MATCH_KEYBIT |
+ INPUT_DEVICE_ID_MATCH_ABSBIT,
.evbit = { BIT(EV_KEY) | BIT(EV_ABS) },
.keybit = { [LONG(BTN_TOUCH)] = BIT(BTN_TOUCH) },
.absbit = { BIT(ABS_X) | BIT(ABS_Y) },
- }, /* A tablet like device, at least touch detection, two absolute axes */
+ }, /* A tablet like device, at least touch detection,
+ two absolute axes */
{
- .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT |
INPUT_DEVICE_ID_MATCH_ABSBIT,
+ .flags = INPUT_DEVICE_ID_MATCH_EVBIT |
+ INPUT_DEVICE_ID_MATCH_KEYBIT |
+ INPUT_DEVICE_ID_MATCH_ABSBIT,
.evbit = { BIT(EV_KEY) | BIT(EV_ABS) },
.keybit = { [LONG(BTN_TOOL_FINGER)] = BIT(BTN_TOOL_FINGER) },
- .absbit = { BIT(ABS_X) | BIT(ABS_Y) | BIT(ABS_PRESSURE) | BIT(ABS_TOOL_WIDTH) },
+ .absbit = { BIT(ABS_X) | BIT(ABS_Y) | BIT(ABS_PRESSURE) |
+ BIT(ABS_TOOL_WIDTH) },
}, /* A touchpad */

{ }, /* Terminating entry */

```

--
--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in

[patch 50/73] Input: mousedev – implement proper locking

the body of a message to majordomo@xxxxxxxxxxxxxxxx

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

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