How should an application ask for uinput module load?

Source: http://linux.derkeiler.com/Mailing−Lists/Kernel/2006−03/msg09678.html

- *From*: Samuel Thibault <samuel.thibault@xxxxxxxxxxxx>
- *Date*: Tue, 28 Mar 2006 21:42:10 +0200

Hi,

Given a freshly booted linux box, hence uinput is not loaded (why would
it be, it doesn't drive any real hardware) ; what is the right way(tm)
for an application to have the uinput module loaded, so that it can open
/dev/input/uinput for emulating keypresses?

− With good−old static /dev, we could just open /dev/input/uinput
(installed by the distribution), and thanks to a
alias char−major−10−223 uinput
line somewhere in /etc/modprobe.d, uinput finally gets auto−loaded.

− With devfs, it doesn't look like it works (/dev/misc/uinput is not
present and opening it just like if it existed doesn't work). But I
read in archives that it could be feasible.

− With udev, this just cannot work. As explained in an earlier thread,
even using a special filesystem that would report the opening attempt
to udevd wouldn't work fine since udevd takes time for creating the
device, and hence the original program needs to be notified ; this
becomes racy.

So what is the correct way to do it? I can see two approaches:

Using modprobe:
− try to use /dev/input/uinput ; if it succeeds, fine.
− else, if errno != ENOENT, fail
− else, (ENOENT)
− try to call `cat /proc/sys/kernel/modprobe` uinput
− try to use /dev/input/uinput again ; if it succeeds, fine
− else, assume that it really wasn't compiled, and hence fail.

Triggering auto−load by creating one's own node.
− try to use /dev/input/uinput ; if it suceeds, fine.
− else, if errno != ENOENT, fail
− else, (ENOENT)
− mknod /somewhere/safe/uinput c 10 223
− use /somewhere/safe/uinput ; if it succeeds, fine

− else, assume that it really wasn't compiled, and hence fail.

I guess the same problem arises for loop devices and all such virtual devices...

Regards,
Samuel
−