NAME

vkbd - the virtual AT keyboard interface

SYNOPSIS

device vkbd

DESCRIPTION

The **vkbd** interface is a software loopback mechanism that can be loosely described as the virtual AT keyboard analog of the pty(4), that is, **vkbd** does for virtual AT keyboards what the pty(4) driver does for terminals.

The **vkbd** driver, like the pty(4) driver, provides two interfaces: a keyboard interface like the usual facility it is simulating (a virtual AT keyboard in the case of **vkbd**, or a terminal for pty(4)), and a character-special device "control" interface.

The virtual AT keyboards are named *vkbd0*, *vkbd1*, etc., one for each control device that has been opened.

The **vkbd** interface permits opens on the special control device */dev/vkbdctl*. When this device is opened, **vkbd** will return a handle for the lowest unused *vkbdctl* device (use devname(3) to determine which).

Each virtual AT keyboard supports the usual keyboard interface ioctl(2)s, and thus can be used with kbdcontrol(1) like any other keyboard. The control device supports exactly the same ioctl(2)s as the virtual AT keyboard device. Writing AT scan codes to the control device generates an input on the virtual AT keyboard, as if the (non-existent) hardware had just received it.

The virtual AT keyboard control device, normally */dev/vkbdctl<N>*, is exclusive-open (it cannot be opened if it is already open) and is restricted to the super-user. A read(2) call will return the virtual AT keyboard status structure (defined in <*dev/vkbd/vkbd_var.h*>) if one is available; if not, it will either block until one is or return EWOULDBLOCK, depending on whether non-blocking I/O has been enabled.

A write(2) call passes AT scan codes to be "received" from the virtual AT keyboard. Each AT scan code must be passed as *unsigned int*. Although AT scan codes must be passes as *unsigned int*s, the size of the buffer passed to write(2) still should be in bytes, i.e.,

```
static unsigned int     codes[] =
{
/*      Make    Break */
```

```
        0x1e,   0x9e
};

int
main(void)
{
        int     fd, len;

        fd = open("/dev/vkbdctl0", O_RDWR);
        if (fd < 0)
                err(1, "open");

        /* Note sizeof(codes) - not 2! */
        len = write(fd, codes, sizeof(codes));
        if (len < 0)
                err(1, "write");

        close(fd);

        return (0);
}
```

Write will block if there is not enough space in the input queue.

The control device also supports select(2) for read and write.

On the last close of the control device, the virtual AT keyboard is removed. All queued scan codes are thrown away.

SEE ALSO

kbdcontrol(1), atkbdc(4), psm(4), syscons(4), vt(4)

HISTORY

The **vkbd** module was implemented in FreeBSD 6.0.

AUTHORS

Maksim Yevmenkin <*m_evmenkin@yahoo.com*>

CAVEATS

The **vkbd** interface is a software loopback mechanism, and, thus ddb(4) will not work with it. Current

implementation of the syscons(4) driver can accept input from only one keyboard, even if it is virtual. Thus it is not possible to have both wired and virtual keyboard to be active at the same time. It is, however, in principal possible to obtain AT scan codes from the different sources and write them into the same virtual keyboard. The virtual keyboard state synchronization is the user’s responsibility.