Keyboard notifier

One can use register_keyboard_notifier to get called back on keyboard events (see kbd_keycode() function for details). The passed structure is keyboard_notifier_param (see <linux/keyboard.h>):

- 'vc' always provide the VC for which the keyboard event applies;
- 'down' is 1 for a key press event, 0 for a key release;
- 'shift' is the current modifier state, mask bit indexes are KG_*;
- 'ledstate' is the current LED state;
- 'value' depends on the type of event.
- KBD_KEYCODE events are always sent before other events, value is the keycode.
- KBD_UNBOUND_KEYCODE events are sent if the keycode is not bound to a keysym. value is the keycode.
- KBD_UNICODE events are sent if the keycode -> keysym translation produced a unicode character. value is the unicode value.
- KBD_KEYSYM events are sent if the keycode -> keysym translation produced a non-unicode character. value is the keysym.
- KBD_POST_KEYSYM events are sent after the treatment of non-unicode keysyms. That permits one to inspect the resulting LEDs for instance.

For each kind of event but the last, the callback may return NOTIFY_STOP in order to "eat" the event: the notify loop is stopped and the keyboard event is dropped.

In a rough C snippet, we have:

```
kbd_keycode(keycode) {
    ...
    params.value = keycode;
    if (notifier_call_chain(KBD_KEYCODE,&params) == NOTIFY_STOP)
        || !bound) {
            notifier_call_chain(KBD_UNBOUND_KEYCODE,&params);
            return;
    }

    if (unicode) {
            param.value = unicode;
            if (notifier_call_chain(KBD_UNICODE,&params) == NOTIFY_STOP)
                    return;
            emit unicode;
            return;
    }

    params.value = keysym;
    if (notifier_call_chain(KBD_KEYSYM,&params) == NOTIFY_STOP)
            return;
    apply keysym;
    notifier_call_chain(KBD_POST_KEYSYM,&params);
}
```

> **Note**
>
> This notifier is usually called from interrupt context.