NAME

keyboard - pc keyboard interface

DESCRIPTION

The PC keyboard is used as the console character input device. The keyboard is owned by the current virtual console. To switch between the virtual consoles use the sequence *ALT+Fn*, which means hold down ALT and press one of the function keys. The virtual console with the same number as the function key is then selected as the current virtual console and given exclusive use of the keyboard and display.

The console allows entering values that are not physically present on the keyboard via a special keysequence. To use this facility press and hold down ALT, then enter a decimal number from 0-255 via the numerical keypad, then release ALT. The entered value is then used as the ASCII value for one character. This way it is possible to enter any ASCII value, not present on the keyboard. The console driver also includes a history function. It is activated by pressing the scroll-lock key. This holds the display, and enables the cursor arrows for scrolling up and down through the last scrolled out lines.

The keyboard is configurable to suit the individual user and the different national layout.

The keys on the keyboard can have any of the following functions:

Normal key	Enter the ASCII value associated with the key.
Function key	Enter a string of ASCII values.
Switch Key	Switch virtual console.
Modifier Key	Change the meaning of another key.

The keyboard is seen as a number of keys numbered from 1 to n. This number is often referred to as the "scancode" for a given key. The number of the key is transmitted as an 8 bit char with bit 7 as 0 when a key is pressed, and the number with bit 7 as 1 when released. This makes it possible to make the mapping of the keys fully configurable.

The meaning of every key is programmable via the PIO_KEYMAP ioctl call, that takes a structure keymap_t as argument. The layout of this structure is as follows:

```
struct keymap {
        u_short    n_keys;
        struct key_t {
                u_char map[NUM_STATES];
                u_char spcl;
                u_char flgs;
```

```
        } key[NUM_KEYS];
};
```

The field n_keys tells the system how many keydefinitions (scancodes) follows. Each scancode is then specified in the key_t substructure.

Each scancode can be translated to any of 8 different values, depending on the shift, control, and alt state. These eight possibilities are represented by the map array, as shown below:

scan code	base	shift	cntrl	shift	alt	alt shift	cntrl	cntrl shift
map[n]	0	1	2	3	4	5	6	7
0x1E	’a’	’A’	0x01	0x01	’a’	’A’	0x01	0x01

This is the default mapping for the key labelled ’A’ which normally has scancode 0x1E. The eight states are as shown, giving the ’A’ key its normal behavior. The spcl field is used to give the key "special" treatment, and is interpreted as follows. Each bit corresponds to one of the states above. If the bit is 0 the key emits the number defined in the corresponding map[] entry. If the bit is 1 the key is "special". This means it does not emit anything; instead it changes the "state". That means it is a shift, control, alt, lock, switch-screen, function-key or no-op key. The bitmap is backwards i.e., 7 for base, 6 for shift etc.

The flgs field defines if the key should react on caps-lock (1), num-lock (2), both (3) or ignore both (0).

The kbdcontrol(1) utility is used to load such a description into/outof the kernel at runtime. This makes it possible to change the key assignments at runtime, or more important to get (GIO_KEYMAP ioctl) the exact key meanings from the kernel (e.g. used by the X server).

The function keys can be programmed using the SETFKEY ioctl call.

This ioctl takes an argument of the type fkeyarg_t:

```
struct fkeyarg {
        u_short   keynum;
        char      keydef[MAXFK];
        char      flen;
};
```

The field keynum defines which function key that is programmed. The array keydef should contain the

new string to be used (MAXFK long), and the length should be entered in flen.

The GETFKEY ioctl call works in a similar manner, except it returns the current setting of keynum.

The function keys are numbered like this:

F1-F12	key 1 - 12
Shift F1-F12	key 13 - 24
Ctrl F1-F12	key 25 - 36
Ctrl+shift F1-F12	key 37 - 48

Home	key 49
Up arrow	key 50
Page Up	key 51
(keypad) -	key 52
Left arrow	key 53
(keypad) 5	key 54
Right arrow	key 55
(keypad) +	key 56
End	key 57
Down arrow	key 58
Page down	key 59
Insert	key 60
Delete	key 61

Left window	key 62
Right window	key 63
Menu	key 64

The kbdcontrol(1) utility also allows changing these values at runtime.

AUTHORS

Sœren Schmidt <*sos@FreeBSD.org*>