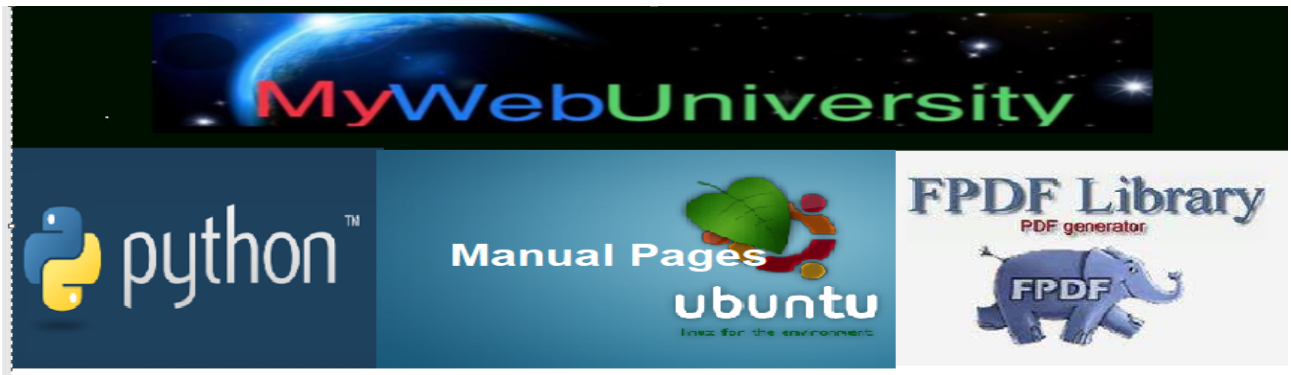

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'ioctl_kd.2'

$ man ioctl_kd.2

ioctl_kd(2) System Calls Manual ioctl_kd(2)

NAME

ioctl_kd - ioctls for console terminal and virtual consoles

SYNOPSIS

#include <linux/kd.h> /* Definition of op constants */

#include <sys/ioctl.h>

int ioctl(int fd, unsigned long op, void *argp);

DESCRIPTION

The following Linux-specific ioctl(2) operations are supported for console terminals and virtual consoles.

KDGETLED

Get state of LEDs. argp points to a uint8_t. The lower three bits of *argp are set to the state of the LEDs, as follows:

LED_CAP 0x04 caps lock led

LED_NUM 0x02 num lock led

LED_SCR 0x01 scroll lock led

KDSETLED

Set the LEDs. The LEDs are set to correspond to the lower three bits of the unsigned long integer in argp. However, if a higher order bit is set, the LEDs revert to nor? mal: displaying the state of the keyboard functions of caps lock, num lock, and scroll lock.

Before Linux 1.1.54, the LEDs just reflected the state of the corresponding keyboard flags, and KDGETLED/KDSETLED would also change the keyboard flags. Since Linux 1.1.54 the LEDs can

be made to display arbitrary information, but by default they display the keyboard flags. The following two ioctls are used to access the keyboard flags.

KDGKBLED

Get keyboard flags CapsLock, NumLock, ScrollLock (not lights). argp points to a uint8_t which is set to the flag state. The low order three bits (mask 0x7) get the current flag state, and the low order bits of the next nibble (mask 0x70) get the de? fault flag state. (Since Linux 1.1.54.)

KDSKBLED

Set keyboard flags CapsLock, NumLock, ScrollLock (not lights). argp is an unsigned long integer that has the desired flag state. The low order three bits (mask 0x7) have the flag state, and the low order bits of the next nibble (mask 0x70) have the default flag state. (Since Linux 1.1.54.)

KDGKBTYPE

Get keyboard type. This returns the value KB_101, defined as 0x02.

KDADDIO

Add I/O port as valid. Equivalent to ioperm(arg,1,1).

KDDELIO

Delete I/O port as valid. Equivalent to ioperm(arg,1,0).

KDENABIO

Enable I/O to video board. Equivalent to ioperm(0x3b4, 0x3df-0x3b4+1, 1).

KDDISABIO

Disable I/O to video board. Equivalent to ioperm(0x3b4, 0x3df-0x3b4+1, 0).

KDSETMODE

Set text/graphics mode. argp is an unsigned integer containing one of:

KD_TEXT 0x00

KD_GRAPHICS 0x01

KDGETMODE

Get text/graphics mode. argp points to an int which is set to one of the values shown above for KDSETMODE.

KDMKTONE

Generate tone of specified length. The lower 16 bits of the unsigned long integer in argp specify the period in clock cycles, and the upper 16 bits give the duration in msec. If the duration is zero, the sound is turned off. Control returns immedi?

ately. For example, argp = (125<<16) + 0x637 would specify the beep normally associ?
ated with a ctrl-G. (Thus, since Linux 0.99pl1; broken in Linux 2.1.49-50.)

KIOCSOUND

Start or stop sound generation. The lower 16 bits of argp specify the period in clock cycles (that is, argp = 1193180/frequency). argp = 0 turns sound off. In ei? ther case, control returns immediately.

GIO_CMAP

Get the current default color map from kernel. argp points to a 48-byte array. (Since Linux 1.3.3.)

PIO_CMAP

Change the default text-mode color map. argp points to a 48-byte array which con? tains, in order, the Red, Green, and Blue values for the 16 available screen colors: 0 is off, and 255 is full intensity. The default colors are, in order: black, dark red, dark green, brown, dark blue, dark purple, dark cyan, light grey, dark grey, bright red, bright green, yellow, bright blue, bright purple, bright cyan, and white. (Since Linux 1.3.3.)

GIO_FONT

Gets 256-character screen font in expanded form. argp points to an 8192-byte array. Fails with error code EINVAL if the currently loaded font is a 512-character font, or if the console is not in text mode.

GIO_FONTX

Gets screen font and associated information. argp points to a struct consolefontdesc (see PIO_FONTX). On call, the charcount field should be set to the maximum number of characters that would fit in the buffer pointed to by chardata. On return, the char? count and charheight are filled with the respective data for the currently loaded font, and the chardata array contains the font data if the initial value of charcount indicated enough space was available; otherwise, the buffer is untouched and errno is set to ENOMEM. (Since Linux 1.3.1.)

PIO_FONT

Sets 256-character screen font. Load font into the EGA/VGA character generator. argp points to an 8192-byte map, with 32 bytes per character. Only the first N of them are used for an 8xN font (0 < N <= 32). This call also invalidates the Unicode mapping.

PIO_FONTX

Sets screen font and associated rendering information. argp points to a

```
struct consolefontdesc {
    unsigned short charcount;  /* characters in font
                                  (256 or 512) */
    unsigned short charheight; /* scan lines per
                                  character (1-32) */
    char          *chardata;   /* font data in
                                  expanded form */
};
```

If necessary, the screen will be appropriately resized, and SIGWINCH sent to the ap? propriate processes. This call also invalidates the Unicode mapping. (Since Linux 1.3.1.)

PIO_FONTRESET

Resets the screen font, size, and Unicode mapping to the bootup defaults. argp is unused, but should be set to NULL to ensure compatibility with future versions of Linux. (Since Linux 1.3.28.)

GIO_SCRNMAP

Get screen mapping from kernel. argp points to an area of size E_TABSZ, which is loaded with the font positions used to display each character. This call is likely to return useless information if the currently loaded font is more than 256 charac? ters.

GIO_UNISCRNMAP

Get full Unicode screen mapping from kernel. argp points to an area of size E_TABSZ*sizeof(unsigned short), which is loaded with the Unicodes each character rep? resent. A special set of Unicodes, starting at U+F000, are used to represent "direct to font" mappings. (Since Linux 1.3.1.)

PIO_SCRNMAP

Loads the "user definable" (fourth) table in the kernel which maps bytes into console screen symbols. argp points to an area of size E_TABSZ.

PIO_UNISCRNMAP

Loads the "user definable" (fourth) table in the kernel which maps bytes into Uni? codes, which are then translated into screen symbols according to the currently

loaded Unicode-to-font map. Special Unicodes starting at U+F000 can be used to map directly to the font symbols. (Since Linux 1.3.1.)

GIO_UNIMAP

Get Unicode-to-font mapping from kernel. argp points to a

```
struct unimapdesc {
    unsigned short  entry_ct;
    struct unipair *entries;
};
```

where entries points to an array of

```
struct unipair {
    unsigned short unicode;
    unsigned short fontpos;
};
```

(Since Linux 1.1.92.)

PIO_UNIMAP

Put unicode-to-font mapping in kernel. argp points to a struct unimapdesc. (Since Linux 1.1.92)

PIO_UNIMAPCLR

Clear table, possibly advise hash algorithm. argp points to a

```
struct unimapinit {
    unsigned short advised_hashsize;  /* 0 if no opinion */
    unsigned short advised_hashstep;  /* 0 if no opinion */
    unsigned short advised_hashlevel; /* 0 if no opinion */
};
```

(Since Linux 1.1.92.)

KDGKBMODE

Gets current keyboard mode. argp points to a long which is set to one of these:

```
K_RAW         0x00  /* Raw (scancode) mode */
K_XLATE       0x01  /* Translate keycodes using keymap */
K_MEDIUMRAW   0x02  /* Medium raw (scancode) mode */
K_UNICODE     0x03  /* Unicode mode */
K_OFF         0x04  /* Disabled mode (since Linux 2.6.39) */
```

KDSKBMODE

Sets current keyboard mode. argp is a long equal to one of the values shown for KDGKBMODE.

KDGKBMETA

Gets meta key handling mode. argp points to a long which is set to one of these:

K_METABIT 0x03 set high order bit

K_ESCPREFIX 0x04 escape prefix

KDSKBMETA

Sets meta key handling mode. argp is a long equal to one of the values shown above for KDGKBMETA.

KDGKBENT

Gets one entry in key translation table (keycode to action code). argp points to a

```
struct kbentry {
    unsigned char  kb_table;
    unsigned char  kb_index;
    unsigned short kb_value;
};
```

with the first two members filled in: kb_table selects the key table (0 <= kb_table < MAX_NR_KEYMAPS), and kb_index is the keycode (0 <= kb_index < NR_KEYS). kb_value is set to the corresponding action code, or K_HOLE if there is no such key, or K_NO? SUCHMAP if kb_table is invalid.

KDSKBENT

Sets one entry in translation table. argp points to a struct kbentry.

KDGKBSENT

Gets one function key string. argp points to a

```
struct kbsentry {
    unsigned char kb_func;
    unsigned char kb_string[512];
};
```

kb_string is set to the (null-terminated) string corresponding to the kb_functh func? tion key action code.

KDSKBSENT

Sets one function key string entry. argp points to a struct kbsentry.

KDGKBDIACR

Read kernel accent table. argp points to a

```
struct kbdiacrs {
    unsigned int   kb_cnt;
    struct kbdiacr kbdiacr[256];
};
```

where kb_cnt is the number of entries in the array, each of which is a

```
struct kbdiacr {
    unsigned char diacr;
    unsigned char base;
    unsigned char result;
};
```

KDGETKEYCODE

Read kernel keycode table entry (scan code to keycode). argp points to a

```
struct kbkeycode {
    unsigned int scancode;
    unsigned int keycode;
};
```

keycode is set to correspond to the given scancode. (89 <= scancode <= 255 only. For 1 <= scancode <= 88, keycode==scancode.) (Since Linux 1.1.63.)

KDSETKEYCODE

Write kernel keycode table entry. argp points to a struct kbkeycode. (Since Linux 1.1.63.)

KDSIGACCEPT

The calling process indicates its willingness to accept the signal argp when it is generated by pressing an appropriate key combination. (1 <= argp <= NSIG). (See spawn_console() in linux/drivers/char/keyboard.c.)

RETURN VALUE

On success, 0 is returned (except where indicated). On failure, -1 is returned, and errno is set to indicate the error.

ERRORS

EINVAL argp is invalid.

STANDARDS

Linux.

SEE ALSO

ioctl(2), ioctl_console(2)