```
printk(KERN_INFO "i2c_test: write (io) successfully\n");
```

in include/linux/kern_levels.h

```
#define KERN_SOH        "\001"          /* ASCII Start Of Header */
#define KERN_SOH_ASCII  '\001'

#define KERN_EMERG      KERN_SOH "0"    /* system is unusable */
#define KERN_ALERT      KERN_SOH "1"    /* action must be taken immediately */
#define KERN_CRIT       KERN_SOH "2"    /* critical conditions */
#define KERN_ERR        KERN_SOH "3"    /* error conditions */
#define KERN_WARNING    KERN_SOH "4"    /* warning conditions */
#define KERN_NOTICE     KERN_SOH "5"    /* normal but significant condition */
#define KERN_INFO       KERN_SOH "6"    /* informational */
#define KERN_DEBUG      KERN_SOH "7"    /* debug-level messages */

#define KERN_DEFAULT    KERN_SOH "d"    /* the default kernel loglevel */
```

==>

printk("\001" "6" "i2c_test: write (io) successfully\n");

处理printk()中string的code如下(in vprintk_emit() function)

```
                int kern_level = printk_get_level(text);        ①

                if (kern_level) {
                        const char *end_of_header = printk_skip_level(text);    ②
                        switch (kern_level) {
                        case '0' ... '7':                       ③
                                if (level == -1)
                                        level = kern_level - '0';

                        case 'd':       /* KERN_DEFAULT */              ⑥
                                lflags |= LOG_PREFIX;
                        }

                        /*
                         * No need to check length here because vscnprintf
                         * put '\0' at the end of the string. Only valid and
                         * newly printed level is detected.
                         */
                        text_len -= end_of_header - text;       ④
                        text = (char *)end_of_header;           ⑤
                }
```

①

这里的text ==> "\001" "6" "i2c_test: write (io) successfully\n"

```
static inline int printk_get_level(const char *buffer)
{
        if (buffer[0] == KERN_SOH_ASCII && buffer[1]) {
                switch (buffer[1]) {
                case '0' ... '7':
                case 'd':       /* KERN_DEFAULT */
                        return buffer[1];
                }
        }

        return 0;
}
```

buffer[0] = '\001'
buffer[1] = '6'
该function返回'6',字符'6',而非数字的6
②
这里的text ==> "\001" "6" "i2c_test: write (io) successfully\n"

```
static inline const char *printk_skip_level(const char *buffer)
{
        if (printk_get_level(buffer))
                return buffer + 2;

        return buffer;
}
```

返回值buffer指向text + 2 ==> "i2c_test: write (io) successfully\n"
③
printk()调用vprintk_emit()的code如下

vprintk_emit(0, -1, NULL, 0, fmt, args);
即level = -1
level = kern_level - '0';
level = '6' - '0' = 6
④
end_of_header ==> "i2c_test: write (io) successfully\n"
text_len = end_of_header - text = 2
⑤
text ==> "i2c_test: write (io) successfully\n"
⑥
如果
printk(KERN_DEFAULT "i2c_test: write (io) successfully\n");

==>
printk("\001" "d" "i2c_test: write (io) successfully\n");
即level不会被修改，还是-1

```
if (level == -1)
        level = default_message_loglevel;
```

即printk(KERN_DEFAULT "i2c_test: write (io) successfully\n");
会被设定为default_message_loglevel指定的log level。