```
$ cat /proc/sys/kernel/printk
4       4       1       7
```

该文件是由kernel/sysctl.c中的如下code生成

```
        {

                .procname       = "printk",

                .data           = &console_loglevel,

                .maxlen         = 4*sizeof(int),

                .mode           = 0644,

                .proc_handler   = proc_dointvec,

        },
```

即该virtual file中的内容对应的是 (&console_loglevel, &console_loglevel + 4*sizeof(int))

in include/linux/printk.h

```
#define console_loglevel (console_printk[0])
#define default_message_loglevel (console_printk[1])
#define minimum_console_loglevel (console_printk[2])
#define default_console_loglevel (console_printk[3])
```

&console_loglevel ==> &console_printk[0]

in kernel/printk/printk.c

```
int console_printk[4] = {
        CONSOLE_LOGLEVEL_DEFAULT,       /* console_loglevel */
        MESSAGE_LOGLEVEL_DEFAULT,       /* default_message_loglevel */
        CONSOLE_LOGLEVEL_MIN,           /* minimum_console_loglevel */
        CONSOLE_LOGLEVEL_DEFAULT,       /* default_console_loglevel */
};
```

cat /proc/sys/kernel/printk也就是显示的console_printk[4] array中的值。

console_loglevel	default_message_loglevel	minimum_console_loglevel	default_console_loglevel
4	4	1	7

- console_loglevel

多个kernel boot parameters就是控制该变量的。
1. debug, make console_loglevel=10
2. quiet, make console_loglevel=4
3. loglevel=8, 0-9

in init/main.c

```
static int __init debug_kernel(char *str)
{
        console_loglevel = CONSOLE_LOGLEVEL_DEBUG;
        return 0;
}



static int __init quiet_kernel(char *str)
{
        console_loglevel = CONSOLE_LOGLEVEL_QUIET;
        return 0;
}



early_param("debug", debug_kernel);
early_param("quiet", quiet_kernel);

static int __init loglevel(char *str)
{
        int newlevel;

        /*

         * Only update loglevel value when a correct setting was passed,

         * to prevent blind crashes (when loglevel being set to 0) that

         * are quite hard to debug

         */
if (get_option(&str, &newlevel)) {
                console_loglevel = newlevel;
                return 0;
        }

        return -EINVAL;
}

early_param("loglevel", loglevel);
```

只有message level < console_loglevel,该message 才会在console中可见。
比如

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

这里KERN_INFO为6,也就是当前console_loglevel必须为 7 以上，该message才会显示。

```
# echo "7" > /proc/sys/kernel/printk
```

"debug" kernel boot parameter把console_loglevel设为10，就是使得所有message都可以显示，因为当前kernel level如下

in include/linux/kern_levels.h

```
#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 */
```

而"quiet"kernel boot parameter把console_loglevel设为4,则只有KERN_ERR及以下的message才会显示，就连KERN_WARNING都不显示了。

kernel message基于console_loglevel的过滤见如下code
in kernel/printk/printk.c

```
static void call_console_drivers(int level, const char *text, size_t len)
{
        struct console *con;

        trace_console(text, len);

        if (level >= console_loglevel && !ignore_loglevel)      ①
                return;

        if (!console_drivers)
                return;

        for_each_console(con) {

        ......

}
```

①
这里的level即来自与message的level
level = msg->level;
如果message leve >= console_loglevel，则不会在console输出。我们通过串口连接到embedded

system上的putty界面就是console。

- default_message_loglevel

在调用printk()时不指定LOG_LEVEL,象如下code

```
1.  printk("i2c_test: write (io) successfully\n");  (A)
```

那么该message就是被指定由default_message_loglevel指定的level值。

```
1.  $ cat /proc/sys/kernel/printk
2.  4       4       1       7
```

在上面的设定中，(A)处的printk()不会在console上由输出。因为该message的log level被指定为4(default_message_loglevel)，而当前的console_loglevel为4,所以不会输出该message.
要输出该message，有如下两种修改设定。

1. cat 5 > /proc/sys/kernel/printk
修改当前的console_loglevel
2. cat 4 3 > /proc/sys/kernel/printk

default_message_loglevel的机理如下：

```
1.  printk(KERN_DEFAULT "i2c_test: write (io) successfully\n");
2.  printk("i2c_test: write (io) successfully\n");
```

这两者是等价的。

这里KERN_DEFAULT就表示采用default_message_loglevel指定的值作为log level。
见《kernel log level in printk()》

- minimum_console_loglevel

该setting好像是专为syslog用的。

- default_console_loglevel

kernel好像没有使用该setting