in kernel/printk/printk.c

printk() --> vprintk_emit()

```
asmlinkage int vprintk_emit(int facility, int level,
                            const char *dict, size_t dictlen,
                            const char *fmt, va_list args)
{
        static int recursion_bug;
        static char textbuf[LOG_LINE_MAX];               ①
        char *text = textbuf;                                    ②
        size_t text_len = 0;
        enum log_flags lflags = 0;
        unsigned long flags;
        int this_cpu;
        int printed_len = 0;
        bool in_sched = false;
        /* cpu currently holding logbuf_lock in this function */
        static volatile unsigned int logbuf_cpu = UINT_MAX;

        if (level == SCHED_MESSAGE_LOGLEVEL) {
                level = -1;
                in_sched = true;
        }

        boot_delay_msec(level);
        printk_delay();

        /* This stops the holder of console_sem just where we want him */
        local_irq_save(flags);
        this_cpu = smp_processor_id();

        /*
         * Ouch, printk recursed into itself!
         */
        if (unlikely(logbuf_cpu == this_cpu)) {
                /*
                 * If a crash is occurring during printk() on this CPU,
                 * then try to get the crash message out but make sure
                 * we can't deadlock. Otherwise just return to avoid the
                 * recursion and return - but flag the recursion so that
                 * it can be printed at the next appropriate moment:
                 */
                if (!oops_in_progress && !lockdep_recursing(current)) {
                        recursion_bug = 1;
                        local_irq_restore(flags);
                        return 0;
                }
                zap_locks();
        }

        lockdep_off();
        raw_spin_lock(&logbuf_lock);
        logbuf_cpu = this_cpu;

        if (unlikely(recursion_bug)) {
                static const char recursion_msg[] =
                        "BUG: recent printk recursion!";

                recursion_bug = 0;
                /* emit KERN_CRIT message */
                printed_len += log_store(0, 2, LOG_PREFIX|LOG_NEWLINE, 0,
                                         NULL, 0, recursion_msg,
                                         strlen(recursion_msg));
        }

        /*
         * The printf needs to come first; we need the syslog
         * prefix which might be passed-in as a parameter.
         */
        text_len = vscnprintf(text, sizeof(textbuf), fmt, args);        ③

        /* mark and strip a trailing newline */
        if (text_len && text[text_len-1] == '\n') {
                text_len--;
                lflags |= LOG_NEWLINE;
        }

......

}
```

①

#define PREFIX_MAX 32

#define LOG_LINE_MAX (1024 - PREFIX_MAX)

textbuf[LOG_LINE_MAX] ==> textbuf[992]

②

text指向textbuf[992]的首部

③

格式化字符串的buffer长度限制了最多sizeof(textbuf)