LINUX内核开发与调试
--打印消息和日志

张银奎

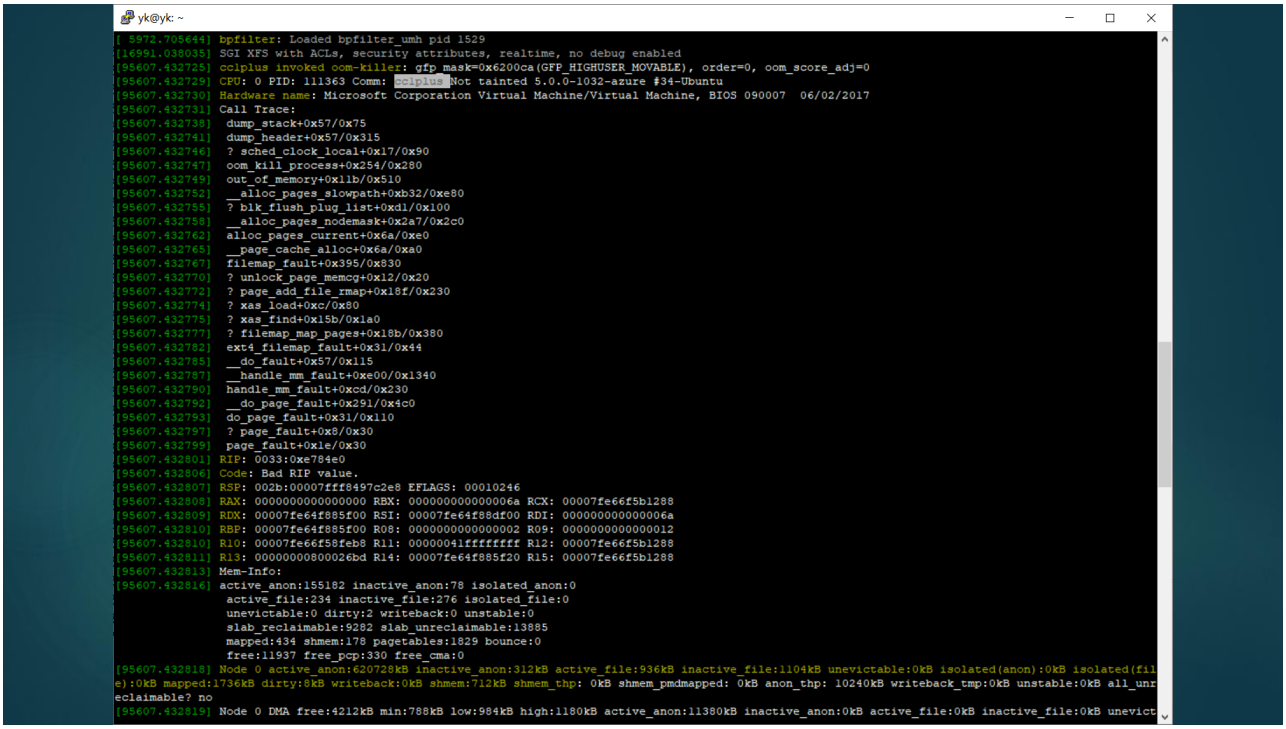
2016/8/20，2018/4/2更新

1

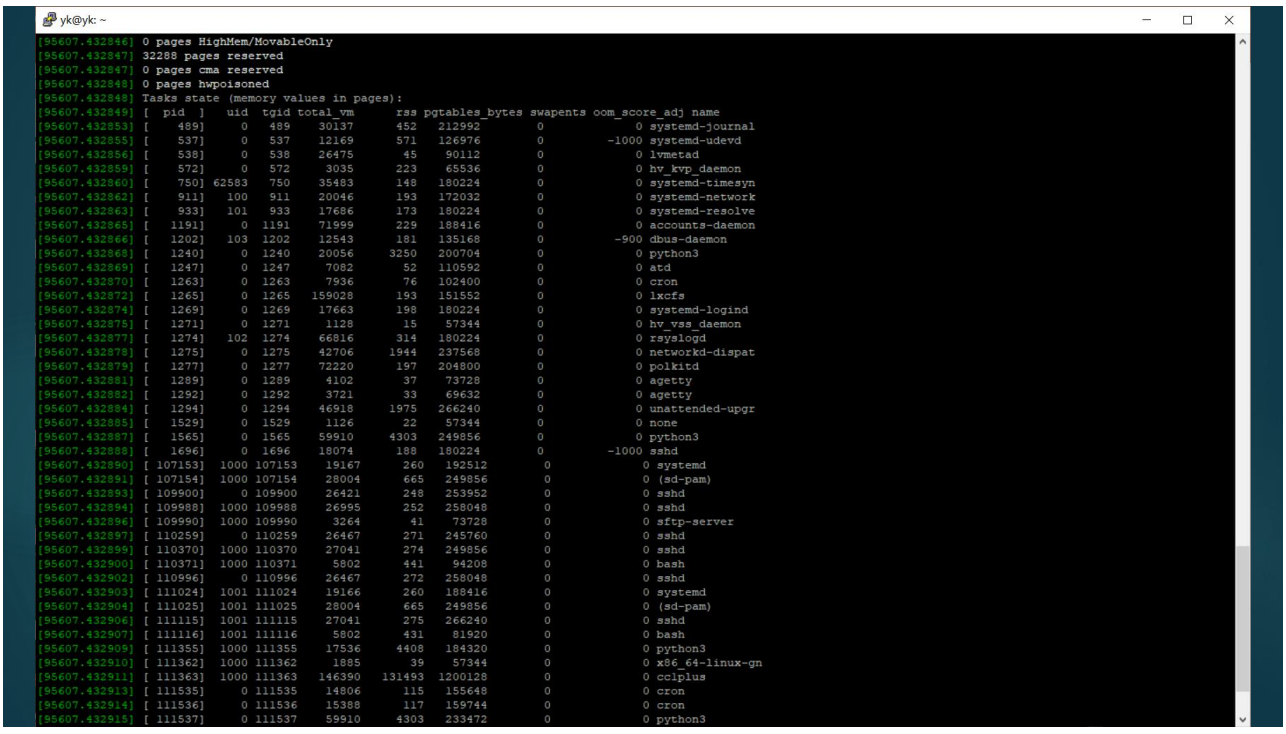
老伙伴报告

```
GGC heuristics: --param ggc-min-expand=91 --param ggc-min-heapsize=114871
ignoring duplicate directory "/usr/include/x86_64-linux-gnu/c++/7"
ignoring nonexistent directory "/usr/local/include/x86_64-linux-gnu"
ignoring nonexistent directory "/usr/lib/gcc/x86_64-linux-gnu/7/../../../../x86_64-linux-gnu/include"
ignoring duplicate directory "/usr/local/lib/python3.6/dist-packages/pybind11/include"
#include "..." search starts here:
#include <...> search starts here:
 spdlog/include/
 /usr/local/lib/python3.6/dist-packages/pybind11/include
 /usr/include/python3.6m
 /usr/include/c++/7
 /usr/include/x86_64-linux-gnu/c++/7
 /usr/include/c++/7/backward
 /usr/lib/gcc/x86_64-linux-gnu/7/include
 /usr/local/include
 /usr/lib/gcc/x86_64-linux-gnu/7/include-fixed
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
GNU C++11 (Ubuntu 7.5.0-3ubuntu1~18.04) version 7.5.0 (x86_64-linux-gnu)
        compiled by GNU C version 7.5.0, GMP version 6.1.2, MPFR version 4.0.1, MPC version 1.1.0, isl ve
rsion isl-0.19-GMP

GGC heuristics: --param ggc-min-expand=91 --param ggc-min-heapsize=114871
Compiler executable checksum: 3eb3dc290cd5714c3e1c3ae751116f07
```

2


yk@yk: ~
[5972.705644] bpfilter: Loaded bpfilter_umh pid 1529
[16991.038035] SGI XFS with ACLs, security attributes, realtime, no debug enabled
[95607.432725] cclplus invoked oom-killer: gfp_mask=0x6200ca(GFP_HIGHUSER_MOVABLE), order=0, oom_score_adj=0
[95607.432729] CPU: 0 PID: 111363 Comm: cclplus Not tainted 5.0.0-1032-azure #34-Ubuntu
[95607.432730] Hardware name: Microsoft Corporation Virtual Machine/Virtual Machine, BIOS 090007 06/02/2017
[95607.432731] Call Trace:
[95607.432738] dump_stack+0x57/0x75
[95607.432741] dump_header+0x57/0x315
[95607.432746] ? sched_clock_local+0x17/0x90
[95607.432747] oom_kill_process+0x254/0x280
[95607.432749] out_of_memory+0x11b/0x510
[95607.432752] __alloc_pages_slowpath+0xb32/0xe80
[95607.432755] ? blk_flush_plug_list+0xd1/0x100
[95607.432758] __alloc_pages_nodemask+0x2a7/0x2c0
[95607.432762] alloc_pages_current+0x6a/0xe0
[95607.432765] __page_cache_alloc+0x6a/0xa0
[95607.432767] filemap_fault+0x395/0x830
[95607.432770] ? unlock_page_memcg+0x12/0x20
[95607.432772] ? page_add_file_rmap+0x18f/0x230
[95607.432774] ? xas_load+0xc/0x80
[95607.432775] ? xas_find+0x15b/0x1a0
[95607.432777] ? filemap_map_pages+0x18b/0x380
[95607.432782] ext4_filemap_fault+0x31/0x44
[95607.432785] __do_fault+0x57/0x115
[95607.432787] __handle_mm_fault+0xe00/0x1340
[95607.432790] handle_mm_fault+0xcd/0x230
[95607.432792] __do_page_fault+0x291/0x4c0
[95607.432793] do_page_fault+0x31/0x110
[95607.432797] ? page_fault+0x8/0x30
[95607.432799] page_fault+0x1e/0x30
[95607.432801] RIP: 0033:0xe784e0
[95607.432806] Code: Bad RIP value.
[95607.432807] RSP: 002b:00007fff8497c2e8 EFLAGS: 00010246
[95607.432808] RAX: 0000000000000000 RBX: 000000000000006a RCX: 00007fe66f5b1288
[95607.432809] RDX: 00007fe64f885f00 RSI: 00007fe64f88df00 RDI: 000000000000006a
[95607.432810] RBP: 00007fe64f885f00 R08: 0000000000000002 R09: 0000000000000012
[95607.432810] R10: 00007fe66f58feb8 R11: 00000041ffffffff R12: 00007fe66f5b1288
[95607.432811] R13: 00000000800026bd R14: 00007fe64f885f20 R15: 00007fe66f5b1288
[95607.432813] Mem-Info:
[95607.432816] active_anon:155182 inactive_anon:78 isolated_anon:0
active_file:234 inactive_file:276 isolated_file:0
unevictable:0 dirty:2 writeback:0 unstable:0
slab_reclaimable:9282 slab_unreclaimable:13885
mapped:434 shmem:178 pagetables:1829 bounce:0
free:11937 free_pcp:330 free_cma:0
[95607.432818] Node 0 active_anon:620728kB inactive_anon:312kB active_file:936kB inactive_file:1104kB unevictable:0kB isolated(anon):0kB isolated(file):0kB mapped:1736kB dirty:8kB writeback:0kB shmem:712kB shmem_thp: 0kB shmem_pmdmapped: 0kB anon_thp: 10240kB writeback_tmp:0kB unstable:0kB all_unreclaimable? no
[95607.432819] Node 0 DMA free:4212kB min:788kB low:984kB high:1180kB active_anon:11380kB inactive_anon:0kB active_file:0kB inactive_file:0kB unevict

3


yk@yk: ~
[95607.432846] 0 pages HighMem/MovableOnly
[95607.432847] 32288 pages reserved
[95607.432847] 0 pages cma reserved
[95607.432848] 0 pages hwpoisoned
[95607.432848] Tasks state (memory values in pages):
[95607.432849] [pid] uid tgid total_vm rss pgtables_bytes swapents oom_score_adj name
[95607.432853] [489] 0 489 30137 452 212992 0 0 systemd-journal
[95607.432855] [537] 0 537 12169 571 126976 0 -1000 systemd-udevd
[95607.432856] [538] 0 538 26475 45 90112 0 0 lvmetad
[95607.432859] [572] 0 572 3035 223 65536 0 0 hv_kvp_daemon
[95607.432860] [750] 62583 750 35483 148 180224 0 0 systemd-timesyn
[95607.432862] [911] 100 911 20046 193 172032 0 0 systemd-network
[95607.432863] [933] 101 933 17686 173 180224 0 0 systemd-resolve
[95607.432865] [1191] 0 1191 71999 229 188416 0 0 accounts-daemon
[95607.432866] [1202] 103 1202 12543 181 135168 0 -900 dbus-daemon
[95607.432868] [1240] 0 1240 20056 3250 200704 0 0 python3
[95607.432869] [1247] 0 1247 7082 52 110592 0 0 atd
[95607.432870] [1263] 0 1263 7936 76 102400 0 0 cron
[95607.432872] [1265] 0 1265 159028 193 151552 0 0 lxcfs
[95607.432874] [1269] 0 1269 17663 198 180224 0 0 systemd-logind
[95607.432875] [1271] 0 1271 1128 15 57344 0 0 hv_vss_daemon
[95607.432877] [1274] 102 1274 66816 314 180224 0 0 rsyslogd
[95607.432878] [1275] 0 1275 42706 1944 237568 0 0 networkd-dispat
[95607.432879] [1277] 0 1277 72220 197 204800 0 0 polkitd
[95607.432881] [1289] 0 1289 4102 37 73728 0 0 agetty
[95607.432882] [1292] 0 1292 3721 33 69632 0 0 agetty
[95607.432884] [1294] 0 1294 46918 1975 266240 0 0 unattended-upgr
[95607.432885] [1529] 0 1529 1126 22 57344 0 0 none
[95607.432887] [1565] 0 1565 59910 4303 249856 0 0 python3
[95607.432888] [1696] 0 1696 18074 188 180224 0 -1000 sshd
[95607.432890] [107153] 1000 107153 19167 260 192512 0 0 systemd
[95607.432891] [107154] 1000 107154 28004 665 249856 0 0 (sd-pam)
[95607.432893] [109900] 0 109900 26421 248 253952 0 0 sshd
[95607.432894] [109988] 1000 109988 26995 252 258048 0 0 sshd
[95607.432896] [109990] 1000 109990 3264 41 73728 0 0 sftp-server
[95607.432897] [110259] 0 110259 26467 271 245760 0 0 sshd
[95607.432899] [110370] 1000 110370 27041 274 249856 0 0 sshd
[95607.432900] [110371] 1000 110371 5802 441 94208 0 0 bash
[95607.432902] [110996] 0 110996 26467 272 258048 0 0 sshd
[95607.432903] [111024] 1001 111024 19166 260 188416 0 0 systemd
[95607.432904] [111025] 1001 111025 28004 665 249856 0 0 (sd-pam)
[95607.432906] [111115] 1001 111115 27041 275 266240 0 0 sshd
[95607.432907] [111116] 1001 111116 5802 431 81920 0 0 bash
[95607.432909] [111355] 1000 111355 17536 4408 184320 0 0 python3
[95607.432910] [111362] 1000 111362 1885 39 57344 0 0 x86_64-linux-gn
[95607.432911] [111363] 1000 111363 146390 131493 1200128 0 0 cclplus
[95607.432913] [111535] 0 111535 14806 115 155648 0 0 cron
[95607.432914] [111536] 0 111536 15388 117 159744 0 0 cron
[95607.432915] [111537] 0 111537 59910 4303 233472 0 0 python3

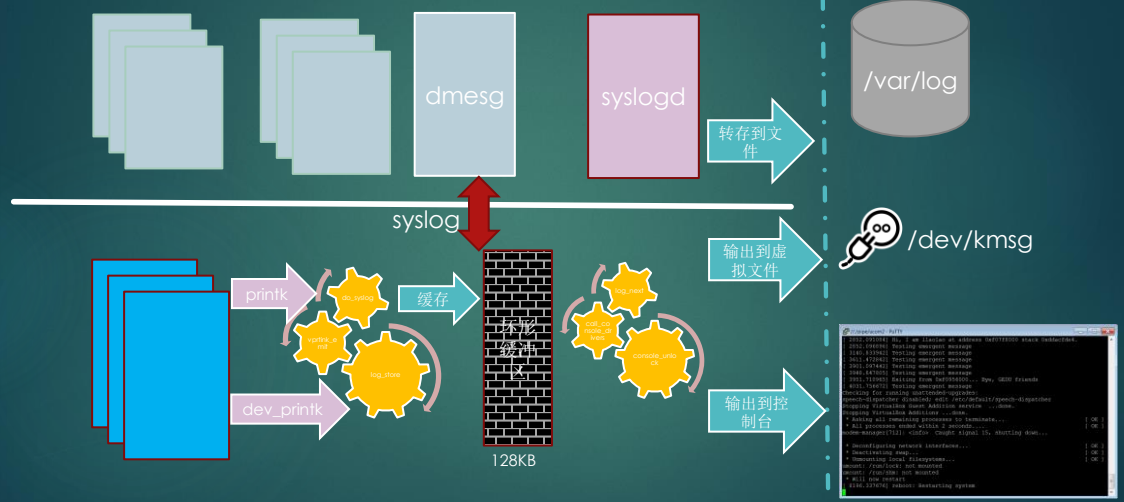
4

Oom-kill

[95607.432916] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/user.slice,task=cc1plus,pid=111363,uid=1000
[95607.432923] Out of memory: Kill process 111363 (cc1plus) score 573 or sacrifice child
[95607.458480] Killed process 111363 (cc1plus) total-vm:585560kB, anon-rss:525972kB, file-rss:0kB, shmem-rss:0kB
[95607.487909] oom_reaper: reaped process 111363 (cc1plus), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
[101352.119097] Adding 2097148k swap on /swapfile. Priority:-2 extents:2 across:2285564k FS

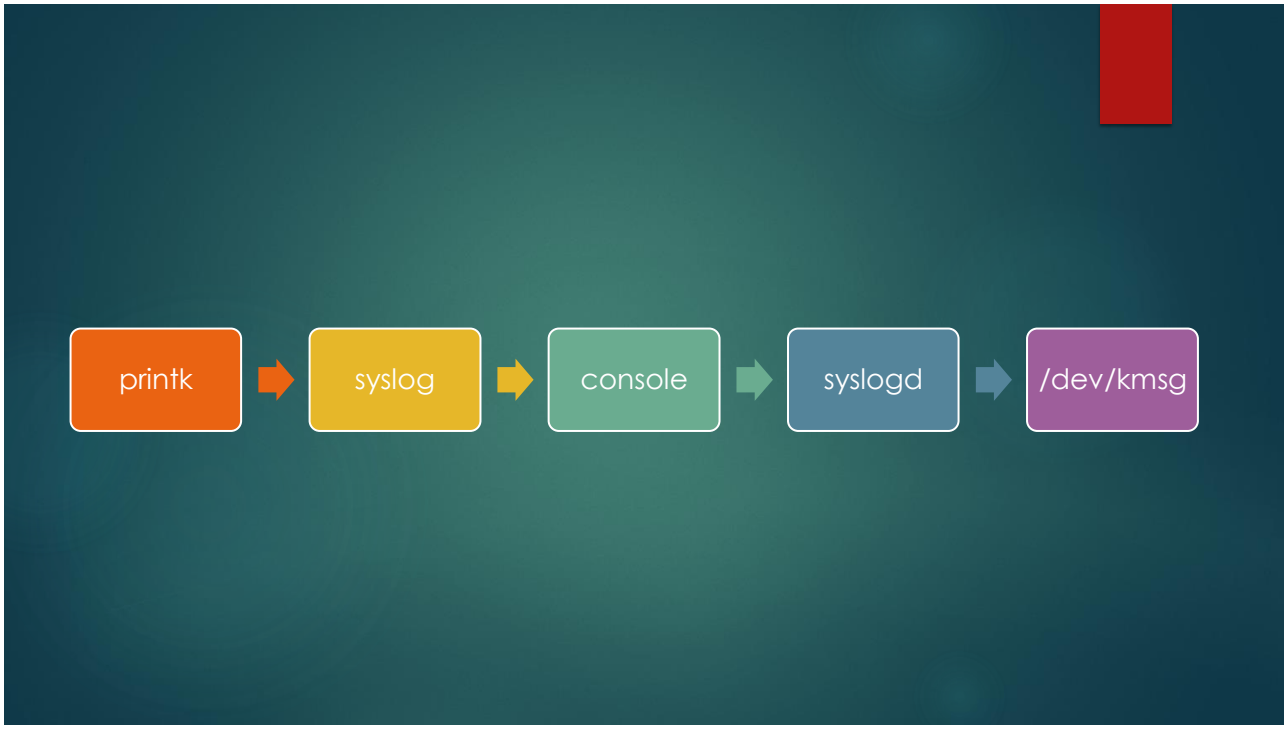
5

系统日志架构



一套设施，两类来源，三类输出

6



7

```
arch/um/drivers/ubd_kern.c:                     printk(KERN_ERR "Failed to initialize ubd device %d :"
arch/um/drivers/ubd_kern.c:             printk(KERN_INFO "ubd: Synchronous mode\n");
arch/um/drivers/ubd_kern.c:             printk(KERN_ERR
arch/um/drivers/ubd_kern.c:             printk(KERN_ERR "um_request_irq failed - errno = %d\n", -err);
arch/um/drivers/ubd_kern.c:                     printk(KERN_ERR "%s: Can't open \"%s\": errno = %d\n",
arch/um/drivers/ubd_kern.c:                     printk("write to io thread failed, "
arch/um/drivers/ubd_kern.c:             printk("do_io - bitmap lseek failed : err = %d\n", -n);
arch/um/drivers/ubd_kern.c:             printk("do_io - bitmap update failed, err = %d fd = %d\n", -n,
arch/um/drivers/ubd_kern.c:                     printk("do_io - sync failed err = %d "
arch/um/drivers/ubd_kern.c:                     printk("do_io - lseek failed : err = %d\n", -err);
arch/um/drivers/ubd_kern.c:                             printk("do_io - read failed, err = %d "
arch/um/drivers/ubd_kern.c:                     printk("do_io - write failed err = %d "
arch/um/drivers/ubd_kern.c:                     printk("io_thread - read failed, fd = %d, "
arch/um/drivers/ubd_kern.c:                     printk("io_thread - short read, fd = %d, "
arch/um/drivers/ubd_kern.c:                     printk("io_thread - write failed, fd = %d, err = %d\n",
arch/um/drivers/daemon_kern.c:  printk("daemon backend (uml_switch version %d) - %s:%s",
arch/um/drivers/daemon_kern.c:  printk("\n");
arch/um/drivers/daemon_kern.c:          printk(KERN_WARNING "daemon_setup : Ignoring data socket "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "new_addr: allocation of sockaddr_un "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : control socket failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : control connect failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : data socket failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : data bind failed, "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "new_addr: allocation of sockaddr_un "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : control setup request "
arch/um/drivers/daemon_user.c:          printk(UM_KERN_ERR "daemon_open : read of data socket failed, "
arch/um/drivers/vde_user.c:             printk(UM_KERN_ERR "vde_user_init: vde_open failed, "
arch/um/drivers/vde_user.c:     printk(UM_KERN_INFO "vde backend - connection opened\n");
arch/um/drivers/vde_user.c:     printk(UM_KERN_WARNING "vde_open - we have no VDECONN to open");
arch/um/drivers/vde_user.c:             printk(UM_KERN_INFO "vde backend - closing connection\n");
arch/um/drivers/vde_user.c:     printk(UM_KERN_WARNING "vde_remove - we have no VDECONN to remove");
arch/um/drivers/vde_user.c:             printk(UM_KERN_ERR "vde_init_libstuff - vde_open_args "
arch/um/drivers/vde_user.c:     args->port ?  printk("port %d", args->port) :
arch/um/drivers/vde_user.c:             printk("undefined port");
```

```
ge@gewubox:~/work/linux-3.12.2$ grep -R "printk" * | wc -l
63170
```

8

printk

```
asmlinkage __visible int printk(const char *fmt, ...)
{
	printk_func_t vprintk_func;
	va_list args;
	int r;

	va_start(args, fmt);

	/*
	 * If a caller overrides the per_cpu printk_func, then it needs
	 * to disable preemption when calling printk(). Otherwise
	 * the printk_func should be set to the default. No need to
	 * disable preemption here.
	 */
	vprintk_func = this_cpu_read(printk_func);
	r = vprintk_func(fmt, args);

	va_end(args);

	return r;
}
```

9

History & Authors

Lister - [D:\bench\linux-4.4.14\kernel\printk\printk.c]

File Edit Options Help

```
/*
 *  linux/kernel/printk.c
 *
 *  Copyright (C) 1991, 1992  Linus Torvalds
 *
 * Modified to make sys_syslog() more flexible: added commands to
 * return the last 4k of kernel messages, regardless of whether
 * they've been read or not.  Added option to suppress kernel printk's
 * to the console.  Added hook for sending the console messages
 * elsewhere, in preparation for a serial line console (someday).
 * Ted Ts'o, 2/11/93.
 * Modified for sysctl support, 1/8/97, Chris Horn.
 * Fixed SMP synchronization, 08/08/99, Manfred Spraul
 *     manfred@colorfullife.com
 * Rewrote bits to get rid of console_lock
 *      01Mar01 Andrew Morton
 */

#include <linux/kernel.h>
```

10

Theodore Ts'o

- Worked on the Linux kernel since 1991, and am probably the first Linux Kernel developer in North America
- the primary developer and maintainer of e2fsprogs, the userspace utilities for the ext2 and ext3 filesystems, and is a maintainer for the ext4 file system
- organizer of the annual Linux Kernel Summit
- https://thunk.org/tytso/

11

格式指示符

Lister - [D:\bench\linux-3.16.3\Documentation\printk-formats.txt]

File Edit Options Help 8 %

```
If variable is of Type,         use printk format specifier:
------------------------------------------------------------
                int                     %d or %x
                unsigned int            %u or %x
                long                    %ld or %lx
                unsigned long           %lu or %lx
                long long               %lld or %llx
                unsigned long long      %llu or %llx
                size_t                  %zu or %zx
                ssize_t                 %zd or %zx

Raw pointer value SHOULD be printed with %p. The kernel supports
the following extended format specifiers for pointer types:

Symbols/Function Pointers:

        %pF     versatile_init+0x0/0x110
        %pf     versatile_init
        %pS     versatile_init+0x0/0x110
        %pSR    versatile_init+0x9/0x110
                (with __builtin_extract_return_addr() translation)
        %ps     versatile_init
        %pB     prev_fn_of_versatile_init+0x88/0x88
```

12

典型执行过程

(gdb) bt

#0 vprintk_emit (facility=0, level=-1, dict=0x0, dictlen=0,
fmt=0xc18b0fa2 "\001\066brd: module loaded\n", args=0xe644bee4 "")
at kernel/printk/printk.c:1497

#1 0xc1670e18 in printk (fmt=0xc18b0fa2 "\001\066brd: module loaded\n")
at kernel/printk/printk.c:1690
#2 0xc1a2f36e in brd_init () at drivers/block/brd.c:623
#3 0xc10020fc in do_one_initcall (fn=0xc1a2f248 <brd_init>) at init/main.c:690
#4 0xc19e6be2 in do_initcall_level (level=<optimized out>) at init/main.c:756
#5 do_initcalls () at init/main.c:764
#6 do_basic_setup () at init/main.c:783
#7 kernel_init_freeable () at init/main.c:885
#8 0xc166b6f0 in kernel_init (unused=<optimized out>) at init/main.c:818
#9 0xc1683037 in ?? () at arch/x86/kernel/entry_32.S:311
#10 0xc166b6e0 in ?? ()

13

发射

asmlinkage int vprintk_emit(int facility, int level,
const char *dict, size_t dictlen,
const char *fmt, va_list args)

- 获取logbuf_lock
- 调用log_store写到环形缓冲区
- 获取console锁，触发输出，唤醒/dev/kmsg和syslog用户

```
/*
 * Try to acquire and then immediately release the console
 * semaphore.  The release will print out buffers and wake up
 * /dev/kmsg and syslog() users.
 */
if (console_trylock_for_printk())
      console_unlock();
```

14

例析

```
vprintk_emit (facility=0, level=-1, dict=0x0, dictlen=0,
   fmt=0xc18b0fa2 "\001\066brd: module loaded\n", args=0xe644bee4 "")
```

- Facility = 0 代表信息源自内核
- Level = -1, 是来自printk的调用时，指定的默认日志级别，vprintk_emit函数内部会尝试从fmt参数中提取level值

```
asmlinkage int vprintk(const char *fmt, va_list args)
{     return vprintk_emit(0, LOGLEVEL_DEFAULT, NULL, 0, fmt, args); }
```

```
#define LOGLEVEL_DEFAULT  -1    /* default (or last) loglevel */
```

15

level of severity

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

- /include/linux/kernel.h

16

消息头结构

```
struct printk_log {
        u64 ts_nsec;        /* timestamp in nanoseconds */
        u16 len;        /* length of entire record */
        u16 text_len;           /* length of text buffer */
        u16 dict_len;           /* length of dictionary buffer */
        u8 facility;        /* syslog facility */
        u8 flags:5;         /* internal record flags */
        u8 level:3;         /* syslog level */
};
```

17

示例

```
*   0000  ff 8f 00 00 00 00 00 00      monotonic time in nsec
*   0008  34 00                        record is 52 bytes long
*   000a        0b 00                  text is 11 bytes long
*   000c              1f 00            dictionary is 23 bytes long
*   000e                    03 00      LOG_KERN (facility) LOG_ERR (level)
*   0010  69 74 27 73 20 61 20 6c      "it's a l"
*         69 6e 65                     "ine"
*   001b           44 45 56 49 43      "DEVIC"
*         45 3d 62 38 3a 32 00 44      "E=b8:2\0D"
*         52 49 56 45 52 3d 62 75      "RIVER=bu"
*         67                           "g"
*   0032     00 00 00                  padding to next message header
*
```

0x31

18

实例

```
(gdb) x/16xb msg
0xc1ae3578: 0x2f    0x69    0x33    0x32    0xa4    0x09    0x00    0x00
0xc1ae3580: 0x30    0x00    0x1f    0x00    0x00    0x00    0x00    0xc6

(gdb) x/32c (void*)msg+16
0xc1ae3588: 97 'a'  116 't' 97 'a'  49 '1'  46 '.'  48 '0'  48 '0'  58 ':'
0xc1ae3590: 32 ' '  99 'c'  111 'o' 110 'n' 102 'f' 105 'i' 103 'g' 117 'u'
0xc1ae3598: 114 'r' 101 'e' 100 'd' 32 ' '  102 'f' 111 'o' 114 'r' 32 ' '
0xc1ae35a0: 85 'U'  68 'D'  77 'M'  65 'A'  47 '/'  51 '3'  51 '3'  0 '\000'
(gdb) x/s (void*)msg+16
0xc1ae3588:  "ata1.00: configured for UDMA/33"
```

19

```
#0  log_store (facility=0, level=6, flags=6, ts_nsec=0, dict=0x0, dict_len=0,
    text=0xc1afd702 "ata1.00: configured for UDMA/33\n", text_len=31)
    at kernel/printk/printk.c:359
#1  0xc10a4948 in vprintk_emit (facility=0, level=6, dict=0x0, dictlen=0,
    fmt=0xc18b97b6 "%sata%u.%02u: %pV", args=0xd17b7d44 "3S\205\301\001")
    at kernel/printk/printk.c:1610
#2  0xc1670e18 in printk (fmt=0xc18b97b6 "%sata%u.%02u: %pV")
    at kernel/printk/printk.c:1690
#3  0xc147eaaf in ata_dev_printk (dev=0xd17b98ac, level=0xc1855333 "\001\066",
    fmt=0xc18b9aa5 "configured for %s%s\n") at drivers/ata/libata-core.c:6846
#4  0xc1484828 in ata_dev_set_mode (dev=0xd17b98ac)
    at drivers/ata/libata-core.c:3270
#5  ata_do_set_mode (link=0xd17b95d8, r_failed_dev=0xd17b7e48)
    at drivers/ata/libata-core.c:3363
#6  0xc148e43d in ata_set_mode (link=0xd17b95d8, r_failed_dev=0xd17b7e48)
    at drivers/ata/libata-eh.c:3232
#7  0xc148f34c in ata_eh_recover (ap=0xd17b8000,
    prereset=0xc1497c50 <piix_pata_prereset>,
    softreset=0xc1492770 <ata_sff_softreset>, hardreset=0,
    postreset=0xc14920f0 <ata_sff_postreset>, r_failed_link=0x0)
    at drivers/ata/libata-eh.c:3844
#8  0xc1490000 in ata_do_eh (ap=0xd17b8000,
    prereset=0xc1497c50 <piix_pata_prereset>,
---Type <return> to continue, or q <return> to quit---
    softreset=0xc1492770 <ata_sff_softreset>, hardreset=0,
```

20

结构化信息 – 属性对(dictionary)

```
* Optionally, a message can carry a dictionary of properties (key/value pairs),
* to provide userspace with a machine-readable message context.
*
* Examples for well-defined, commonly used property names are:
*   DEVICE=b12:8            device identifier
*                           b12:8         block dev_t
*                           c127:3        char dev_t
*                           n8            netdev ifindex
*                           +sound:card0  subsystem:devname
*   SUBSYSTEM=pci           driver-core subsystem name
*
* Valid characters in property names are [a-zA-Z0-9.-_]. The plain text value
* follows directly after a '=' character. Every property is terminated by
* a '\0' character. The last property is not terminated.
```

21

历史

22

Challenges in the past

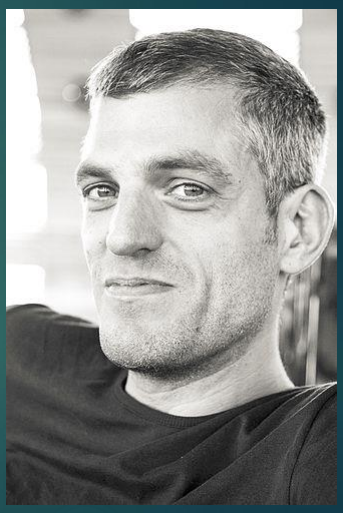
Linux Technology Center

- Martin Schwidefsky, et al., “kernel message catalog”, 2007～2008
- Kazuo Ito, et al., “Kernel Messaging ID”, LinuxCon Japan, 2010
  - As part of OSS Message Pedia (explained later)
- Kay Sievers and Lennart Poettering, “Structured error logging”, Kernel Summit, 2011

23

Kay Sievers

- Best known for developing the udev device manager
- 目前工作在Red Hat，曾经工作在Novell
- 德国柏林
- 为printk增加了结构化消息支持



24

printk: support structured and multi-facility log messages

From: Kay Sievers <kay@vrfy.org>
To: linux-kernel@vger.kernel.org
Subject: [PATCH] printk: support structured and multi-facility log messages
Date: Wed, 04 Apr 2012 21:59:14 +0200
Message-ID: <1333569554.864.3.camel@mop>
Cc: Greg Kroah-Hartmann <greg@kroah.com>
Archive-link: Article

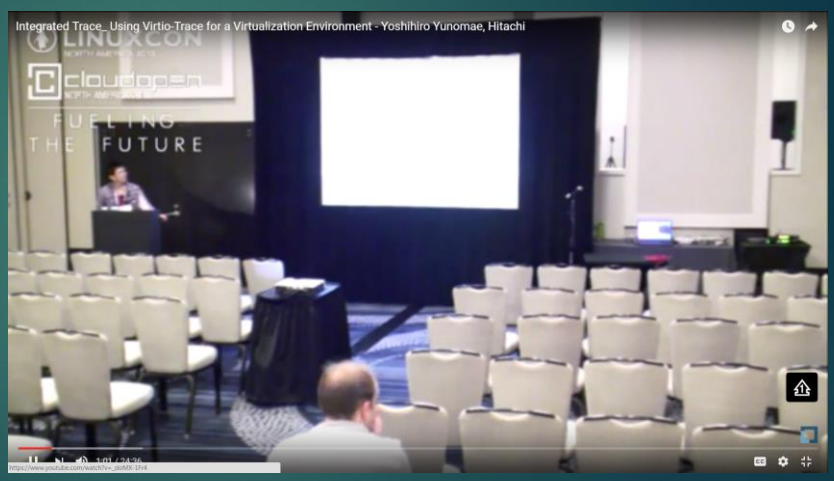
```
From: Kay Sievers <kay@vrfy.org>
Subject: printk: support structured and multi-facility log messages

Kernel log messages are the primary source of information about the overall
state of the system and connected devices. Traditional kernel messages are
mostly human language, targeted at a human reading them. This part of the
picture works very well since a very long time.

However, most machines run unattended almost all of their time, and
software, and not humans, need to process the kernel messages. Having
a machine making sense out of human language messages is inefficient,
unreliable, and sometimes plain impossible to get right. With human
language messages all useful information about their context,
available at the time of creation of the messages, is just thrown
away. Later, software consumers of the messages will need to apply
magic to reconstruct what the context might have been, to be able to
interpret the messages.

This patch extends printk() to be able to attach arbitrary key/value
pairs to logged messages, to carry machine-readable data which
describes the context of the log message at time of its
creation. Users of the log can retrieve, along with the human-readable
message, a key/value dictionary to reliably identify specific devices,
drivers, subsystems, classes and types of messages.
```

- https://lwn.net/Articles/490690/

25

Yoshihiro Yunomae

- Integrated Trace_ Using Virtio-Trace for a Virtualization Environment - Yoshihiro Yunomae, Hitachi
- Hitachi April 2010 – October 2014 (4 years 7 months)
- Akatsuki, Inc.
- October 2014 – Present (1 year 11 months)



26

Subject [PATCH 1/2] printk: Add dictionary information in structure cont
From **Yoshihiro YUNOMAE** <>
Date Fri, 20 Dec 2013 18:41:37 +0900
share 0
share 0
Add dictionary information in structure cont.
Dictionary information is added when a driver uses structured printk, and the information is shown in /dev/kmsg. Current kernel directly stores the information to log_buf. This patch stores the dict information in structure cont first, then the information in cont is stored to log_buf.

Signed-off-by: Yoshihiro YUNOMAE <yoshihiro.yunomae.ez@hitachi.com>
Cc: Kay Sievers <kay@vrfy.org>
Cc: Andrew Morton <akpm@linux-foundation.org>
Cc: Joe Perches <joe@perches.com>
Cc: Tejun Heo <tj@kernel.org>
Cc: Frederic Weisbecker <fweisbec@gmail.com>
Cc: linux-kernel@vger.kernel.org
---
 kernel/printk/printk.c | 70 ++++++++++++++++++++++++++++++++-----------------
 1 file changed, 47 insertions(+), 23 deletions(-)
diff --git a/kernel/printk/printk.c b/kernel/printk/printk.c

- https://lkml.org/lkml/2013/12/20/103

27

vprintk_emit

```
ge@gewubox:~/work/linux-3.12.2$ grep -R "vprintk_emit" *
drivers/usb/storage/debug.c:      r = dev_vprintk_emit(7, &us->pusb_dev->dev, fmt, args);
drivers/base/core.c:int dev_vprintk_emit(int level, const struct device *dev,
drivers/base/core.c:return vprintk_emit(0, level, hdrlen ? hdr : NULL, hdrlen, fmt, args);
drivers/base/core.c:EXPORT_SYMBOL(dev_vprintk_emit);
drivers/base/core.c:r = dev_vprintk_emit(level, dev, fmt, args);
include/linux/device.h:int dev_vprintk_emit(int level, const struct device *dev,
include/linux/device.h:int dev_vprintk_emit(int level, const struct device *dev,
include/linux/printk.h:int vprintk_emit(int facility, int level,
kernel/printk/printk.c:asmlinkage int vprintk_emit(int facility, int level,
kernel/printk/printk.c:EXPORT_SYMBOL(vprintk_emit);
```

28

/linux/device.h

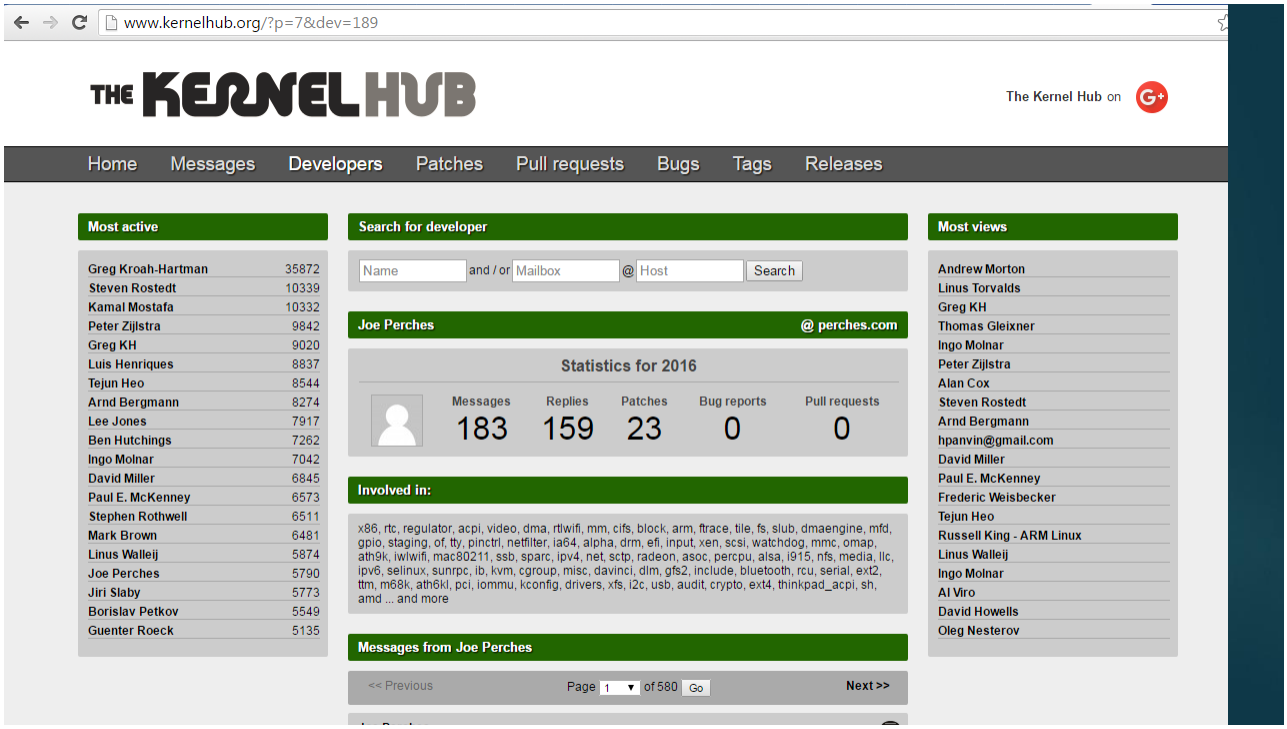
```
int dev_vprintk_emit(int level, const struct device *dev,
                     const char *fmt, va_list args);
int dev_printk_emit(int level, const struct device *dev, const char *fmt, ...);
int dev_printk(const char *level, const struct device *dev,
               const char *fmt, ...);
int dev_emerg(const struct device *dev, const char *fmt, ...);
int dev_alert(const struct device *dev, const char *fmt, ...);
int dev_crit(const struct device *dev, const char *fmt, ...);
int dev_err(const struct device *dev, const char *fmt, ...);
int dev_warn(const struct device *dev, const char *fmt, ...);
int dev_notice(const struct device *dev, const char *fmt, ...);
int _dev_info(const struct device *dev, const char *fmt, ...);
```

29

例子

```
int usb_stor_dbg(const struct us_data *us, const char *fmt, ...)
{
        va_list args;
        int r;

        va_start(args, fmt);

        r = dev_vprintk_emit(7, &us->pusb_dev->dev, fmt, args);

        va_end(args);

        return r;
}
EXPORT_SYMBOL_GPL(usb_stor_dbg);
```

30


www.kernelhub.org/?p=7&dev=189
THE KERNEL HUB
The Kernel Hub on
Home
Messages
Developers
Patches
Pull requests
Bugs
Tags
Releases
Most active
Greg Kroah-Hartman 35872
Steven Rostedt 10339
Kamal Mostafa 10332
Peter Zijlstra 9842
Greg KH 9020
Luis Henriques 8837
Tejun Heo 8544
Arnd Bergmann 8274
Lee Jones 7917
Ben Hutchings 7262
Ingo Molnar 7042
David Miller 6845
Paul E. McKenney 6573
Stephen Rothwell 6511
Mark Brown 6481
Linus Walleij 5874
Joe Perches 5790
Jiri Slaby 5773
Borislav Petkov 5549
Guenter Roeck 5135
Search for developer
Name
and / or
Mailbox
@
Host
Search
Joe Perches
@ perches.com
Statistics for 2016
Messages 183
Replies 159
Patches 23
Bug reports 0
Pull requests 0
Involved in:
x86, rtc, regulator, acpi, video, dma, rtlwifi, mm, cifs, block, arm, ftrace, tile, fs, slub, dmaengine, mfd, gpio, staging, of, tty, pinctrl, netfilter, ia64, alpha, drm, efi, input, xen, scsi, watchdog, mmc, omap, ath9k, iwlwifi, mac80211, ssb, sparc, ipv4, net, sctp, radeon, asoc, percpu, alsa, i915, nfs, media, llc, ipv6, selinux, sunrpc, ib, kvm, cgroup, misc, davinci, dlm, gfs2, include, bluetooth, rcu, serial, ext2, ttm, m68k, ath6kl, pci, iommu, kconfig, drivers, xfs, i2c, usb, audit, crypto, ext4, thinkpad_acpi, sh, amd ... and more
Messages from Joe Perches
<< Previous
Page 1 of 580 Go
Next >>
Most views
Andrew Morton
Linus Torvalds
Greg KH
Thomas Gleixner
Ingo Molnar
Peter Zijlstra
Alan Cox
Steven Rostedt
Arnd Bergmann
hpanvin@gmail.com
David Miller
Paul E. McKenney
Frederic Weisbecker
Tejun Heo
Russell King - ARM Linux
Linus Walleij
Ingo Molnar
Al Viro
David Howells
Oleg Nesterov

31


持续改进
THE KERNEL HUB
printk
<< Previous
Page 1 of 62 Go
Next >>
Byungchul Park
[PATCH v6 2/2] printk: Make printing of spin_dump() deferred to avoid a deadlock
Byungchul Park
[PATCH v6 1/2] printk: Factor out buffering and irq work queuing in printk_deferred
Ivan Delalande
[PATCH] printk: add clear_idx symbol to vmcoreinfo
Sergey Senozhatsky
[RFC][PATCH v2 2/2] printk: Skip messages on oops
Sergey Senozhatsky
[RFC][PATCH v2 1/2] printk: Make printk() completely async
Sergey Senozhatsky
[RFC][PATCH v2 0/2] printk: Make printk() completely async
Jan Kara
[PATCH 2/3] printk: Skip messages on oops
Jan Kara
[PATCH 3/3] printk: debug: Slow down printing to 9600 bauds
Jan Kara
[PATCH 1/3] printk: Make printk() completely async
Sergey Senozhatsky
[PATCH] printk/nmi: restore printk_func in nmi_panic
<< Previous
Page 1 of 62 Go
Next >>
Execution time: 7.2722 seconds

32

config BOOT_PRINTK_DELAY

depends on DEBUG_KERNEL && PRINTK && GENERIC_CALIBRATE_DELAY
help
This build option allows you to read kernel boot messages
by inserting a short delay after each one. The delay is
specified in milliseconds on the kernel command line,
using "**boot_delay=N**".

Ubuntu官方构建已经打开

It is likely that you would also need to use "lpj=M" to preset
the "loops per jiffie" value.
See a previous boot log for the "lpj" value to use for your
system, and then set "lpj=M" before setting "boot_delay=N".
NOTE: Using this option may adversely affect SMP systems.
I.e., processors other than the first one may not boot up.
BOOT_PRINTK_DELAY also may cause LOCKUP_DETECTOR to detect
what it believes to be lockup conditions.

$ cat /boot/config-4.8.0-36-generic | grep "BOOT_PRINTK_DE"
CONFIG_BOOT_PRINTK_DELAY=y

33

动态printk

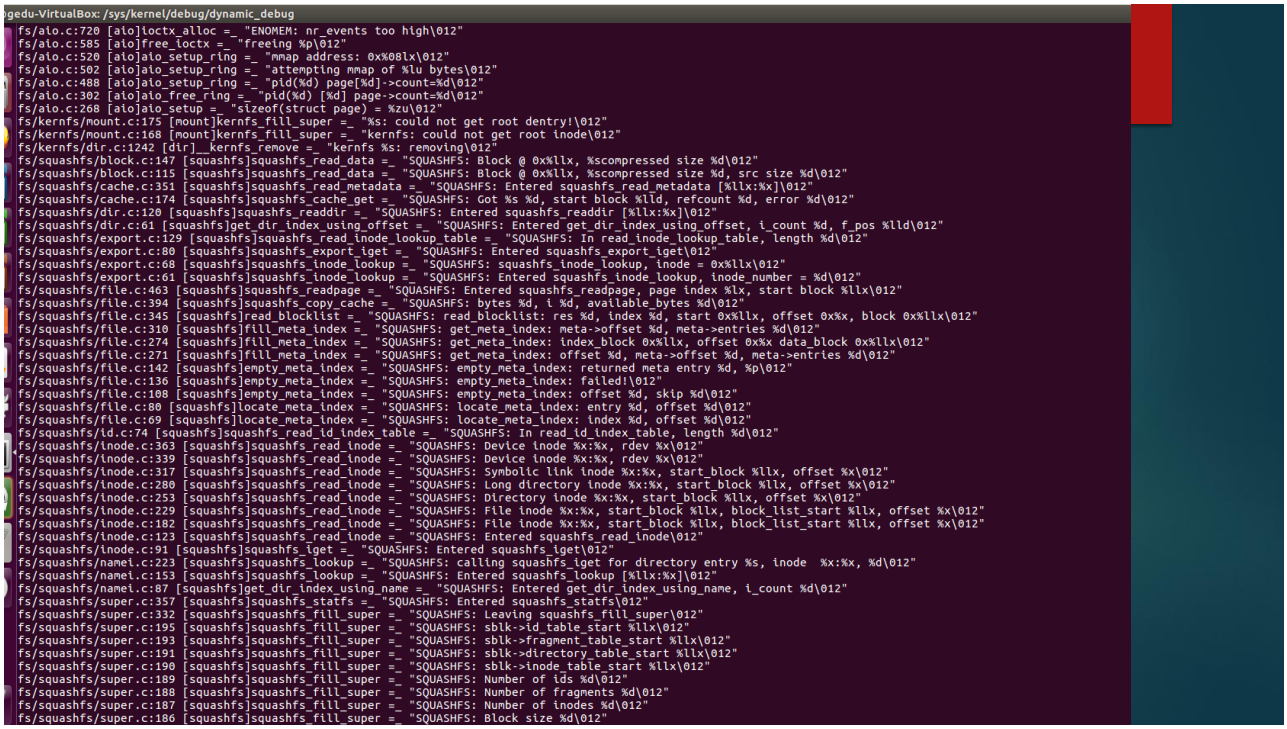
config DYNAMIC_DEBUG
bool "Enable dynamic printk() support"
default n
depends on PRINTK
depends on DEBUG_FS

- 打开编译选项CONFIG_DYNAMIC_DEBUG
  $ cat /boot/config-4.8.0-36-generic | grep "DYNAMIC_DEBUG"
  CONFIG_DYNAMIC_DEBUG=y
- 通过<debugfs>/dynamic_debug/control开关消息

```
// enable all 12 messages in the function svc_process()
nullarbor:~ # echo -n 'func svc_process +p' >
            <debugfs>/dynamic_debug/control
```

- Documentation/dynamic-debug-howto.txt

34



35

```
// enable the message at line 1603 of file svcsock.c
nullarbor:~ # echo -n 'file svcsock.c line 1603 +p' >
                    <debugfs>/dynamic_debug/control

// enable all the messages in file svcsock.c
nullarbor:~ # echo -n 'file svcsock.c +p' >
                    <debugfs>/dynamic_debug/control

// enable all the messages in the NFS server module
nullarbor:~ # echo -n 'module nfsd +p' >
                    <debugfs>/dynamic_debug/control

// enable all 12 messages in the function svc_process()
nullarbor:~ # echo -n 'func svc_process +p' >
                    <debugfs>/dynamic_debug/control

// disable all 12 messages in the function svc_process()
nullarbor:~ # echo -n 'func svc_process -p' >
                    <debugfs>/dynamic_debug/control
```

36

The flags specification comprises a change operation followed by one or more flag characters. The change operation is one of the characters:

- `-` remove the given flags
- `+` add the given flags
- `=` set the flags to the given flags

The flags are:

- `p` enables the pr_debug() callsite.
- `f` Include the function name in the printed message
- `l` Include line number in the printed message
- `m` Include module name in the printed message
- `t` Include thread ID in messages not generated from interrupt context
- `_` No flags are set. (Or'd with others on input)

37

XHCI驱动的例子

```
xhci_dbg(xhci, "%s Endpoint %02d Context (ep_index %02d):\n",
            usb_endpoint_out(epaddr) ? "OUT" : "IN",
            epaddr & USB_ENDPOINT_NUMBER_MASK, i);
```

```
drivers/usb/host/xhci-dbg.c:530 [xhci_hcd]xhci_dbg_ep_ctx =_ "%s Endpoint %02d Context
(ep_index %02d):\012"
```

```
 #define xhci_dbg(xhci, fmt, args...) \
       dev_dbg(xhci_to_hcd(xhci)->self.controller , fmt , ## args)
```

38

```
struct _ddebug {
        /*
         * These fields are used to drive the user interface
         * for selecting and displaying debug callsites.
         */
        const char *modname;
        const char *function;
        const char *filename;
        const char *format;
        unsigned int lineno:18;
        /*
         * The flags field controls the behaviour at the callsite.
         * The bits here are changed dynamically when the user
         * writes commands to <debugfs>/dynamic_debug/control
         */
#define _DPRINTK_FLAGS_NONE         0
#define _DPRINTK_FLAGS_PRINT (1<<0) /* printk() a message using the format */
#define _DPRINTK_FLAGS_INCL_MODNAME       (1<<1)
#define _DPRINTK_FLAGS_INCL_FUNCNAME      (1<<2)
#define _DPRINTK_FLAGS_INCL_LINENO      (1<<3)
#define _DPRINTK_FLAGS_INCL_TID           (1<<4)
#if defined DEBUG
#define _DPRINTK_FLAGS_DEFAULT _DPRINTK_FLAGS_PRINT
#else
#define _DPRINTK_FLAGS_DEFAULT 0
#endif
        unsigned int flags:8;
} __attribute__((aligned(8)));
```

每个可控制的打印点都有这样一个结构体
全部启用，内核文件会增大~2%

要加DEBUG标志编译才产生(见下一页)

39

pr_debug()

Some files call pr_debug(), which is ordinarily an empty macro that discards its arguments at compile time. To enable debugging output, build the appropriate file with -DDEBUG by adding

CFLAGS_[filename].o := -DDEBUG

to the makefile.

For example, to see all attempts to spawn a usermode helper (such as /sbin/hotplug), add to lib/Makefile the line:

CFLAGS_kobject_uevent.o := -DDEBUG

Then boot the new kernel, do something that spawns a usermode helper, and use the "dmesg" command to view the pr_debug() output.

40

公牛格式

From Richard Weinberger <>
Subject [PATCH 0/2] cowsay support
Date Sun, 1 Apr 2018 10:56:20 +0200

share 0 share 0

```
As the printk subsystem gains more and more popularity it is time to add
the final missing feature to make it perfect, namely cowsay output.
Currently only one type of cow is supported and the only user is kmsg,
but I can see more use cases for it and the need for different cow types.

Example of /dev/kmsg usage:
[    1.309561]  ________________________________________
[    1.309561] < systemd[1]: Detected architecture x86-64. >
[    1.309561]  ----------------------------------------
[    1.309561]                     \   ^__^
[    1.309561]                      \  (oo)\_______
[    1.309561]                         (__)\       )\/\
[    1.309561]                             ||----w |
[    1.309561]                             ||     ||

Richard Weinberger (2):
  lib: vsprintf: Implement %pCOW
  printk: Use %pCOW for kmsg

 kernel/printk/printk.c |  5 ++++-
 lib/vsprintf.c         | 52 ++++++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 56 insertions(+), 1 deletion(-)

--
2.13.6
```

https://lkml.org/lkml/2018/4/1/25

41

42

syslog

- Syslog was developed in the 1980s by Eric Allman as part of the Sendmail project
- RFC 3164 - The BSD syslog Protocol
- Standardized by RFC 5424 - The Syslog Protocol
  - https://tools.ietf.org/html/rfc5424

Numerical Code	Severity
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

Table 2. syslog Message Severities

43

Numerical Code	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages (note 1)

	messages generated internally by syslogd
	line printer subsystem
	network news subsystem
	UUCP subsystem
	clock daemon (note 2)
	security/authorization messages (note 1)
	FTP daemon
	NTP subsystem
	log audit (note 1)
	log alert (note 1)
	clock daemon (note 2)
	local use 0 (local0)
	local use 1 (local1)
	local use 2 (local2)
	local use 3 (local3)
	local use 4 (local4)
	local use 5 (local5)
	local use 6 (local6)
	local use 7 (local7)

1. syslog Message Facilities

Diagram 1 shows the different entities separated by layer.

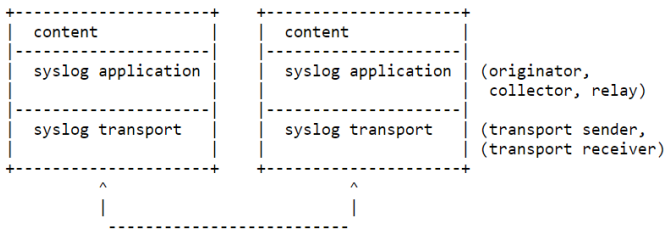


Diagram 1. Syslog Layers

44

Syslog in Linux

```
#include <syslog.h>

    void openlog(const char *ident, int option, int facility);
    void syslog(int priority, const char *format, ...);
    void closelog(void);


SYSCALL_DEFINE3(syslog, int, type, char __user *, buf, int, len)
{
     return do_syslog(type, buf, len, SYSLOG_FROM_READER);
}
```

45

man syslog

```
SYSLOG(3)                                          Linux Programmer's Manual

NAME
       closelog, openlog, syslog, vsyslog - send messages to the system logger

SYNOPSIS
       #include <syslog.h>

       void openlog(const char *ident, int option, int facility);
       void syslog(int priority, const char *format, ...);
       void closelog(void);

       #include <stdarg.h>

       void vsyslog(int priority, const char *format, va_list ap);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       vsyslog(): _BSD_SOURCE

DESCRIPTION
       closelog() closes the descriptor being used to write to the system logger.

       openlog() opens a connection to the system logger for a program.  The strin
       sage, and is typically set to the program name.  The option argument specif
       log()  and  subsequent  calls  to syslog().  The facility argument establis
       subsequent calls to syslog().  Values for option and facility are given bel
       automatically be called by syslog() if necessary, in which case ident will
```

46

Facility

LOG_AUTH security/authorization messages (DEPRECATED Use LOG_AUTHPRIV instead)
LOG_AUTHPRIV security/authorization messages (private)
LOG_CRON clock daemon (cron and at)
LOG_DAEMON system daemons without separate facility value
LOG_FTP ftp daemon
LOG_KERN kernel messages (these can't be generated from user pro - cesses)
LOG_LOCAL0 through LOG_LOCAL7
reserved for local use
LOG_LPR line printer subsystem
LOG_MAIL mail subsystem
LOG_NEWS USENET news subsystem
LOG_SYSLOG messages generated internally by syslogd(8)
LOG_USER (default)
generic user-level messages
LOG_UUCP UUCP subsystem

47

```
samples/kprobes/kretprobe_example.c: * if syslogd is configured to eliminate duplicate messages.)
samples/kprobes/jprobe_example.c: * (Some messages may be suppressed if syslogd is configured to
Binary file scripts/selinux/mdp/mdp matches
Binary file scripts/selinux/genheaders/genheaders matches
security/security.c:int security_syslog(int type)
security/security.c:    return security_ops->syslog(type);
security/Kconfig:       bool "Restrict unprivileged access to the kernel syslog"
security/Kconfig:         syslog via dmesg(8).
security/selinux/hooks.c:#include <linux/syslog.h>
security/selinux/hooks.c:static int selinux_syslog(int type)
security/selinux/hooks.c:       .syslog =                       selinux_syslog,
security/selinux/include/classmap.h:      { "ipc_info", "syslog_read", "syslog_mod",
security/selinux/include/classmap.h:        "syslog_console", "module_request", NULL } },
security/selinux/include/classmap.h:      { "mac_override", "mac_admin", "syslog", "wake_alarm", "block_suspend",
security/capability.c:static int cap_syslog(int type)
security/capability.c:  set_to_cap_if_null(ops, syslog);
security/smack/smack_lsm.c: * smack_syslog - Smack approval on syslog
security/smack/smack_lsm.c:static int smack_syslog(int typefrom_file)
security/smack/smack_lsm.c:     .syslog =                       smack_syslog,
tools/usb/testusb.c:            /* FIXME need a "syslog it" option for background testing */
tools/hv/hv_kvp_daemon.c:#include <syslog.h>
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to acquire the lock pool: %d; error: %d %s", pool,
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to release the lock pool: %d; error: %d %s", pool,
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to open file, pool: %d; error: %d %s", pool,
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to write file, pool: %d", pool);
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to open file, pool: %d; error: %d %s", pool,
tools/hv/hv_kvp_daemon.c:                       syslog(LOG_ERR, "Failed to read file, pool: %d", pool);
tools/hv/hv_kvp_daemon.c:                               syslog(LOG_ERR, "malloc failed");
tools/hv/hv_kvp_daemon.c:                       syslog(LOG_ERR, "Failed to create '%s'; error: %d %s", KVP_CONFIG_LOC,
tools/hv/hv_kvp_daemon.c:                               syslog(LOG_ERR, "Failed to read file, pool: %d",
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to open config file; error: %d %s",
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to execute cmd '%s'; error: %d %s",
tools/hv/hv_kvp_daemon.c:       syslog(LOG_ERR, "Failed to write config file");
tools/hv/hv_kvp_daemon.c:       syslog(LOG_INFO, "KVP starting; pid is:%d", getpid());
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to allocate netlink buffers");
tools/hv/hv_kvp_daemon.c:               syslog(LOG_ERR, "Failed to initialize the pools");
```

48

logger

```
ge@gewubox:~/work/linux-3.12.2$ logger -t GEDU this is a test by Raymond
```

```
logger -p kern.emerg -t GEDU this is a test message
```

tail –f /var/log/syslog

```
Aug 21 18:13:56 gewubox -- MARK --
Aug 21 18:17:01 gewubox CRON[3341]: (root) CMD (   cd / && run-parts --report /
tc/cron.hourly)
Aug 21 18:33:56 gewubox -- MARK --
Aug 21 18:53:56 gewubox -- MARK --
Aug 21 19:02:58 gewubox this: is a test by Raymond
Aug 21 19:03:00 gewubox this: is a test by Raymond
Aug 21 19:03:59 gewubox this: is a test by Raymond
Aug 21 19:06:15 gewubox GEDU: this is a test by Raymond
```

49

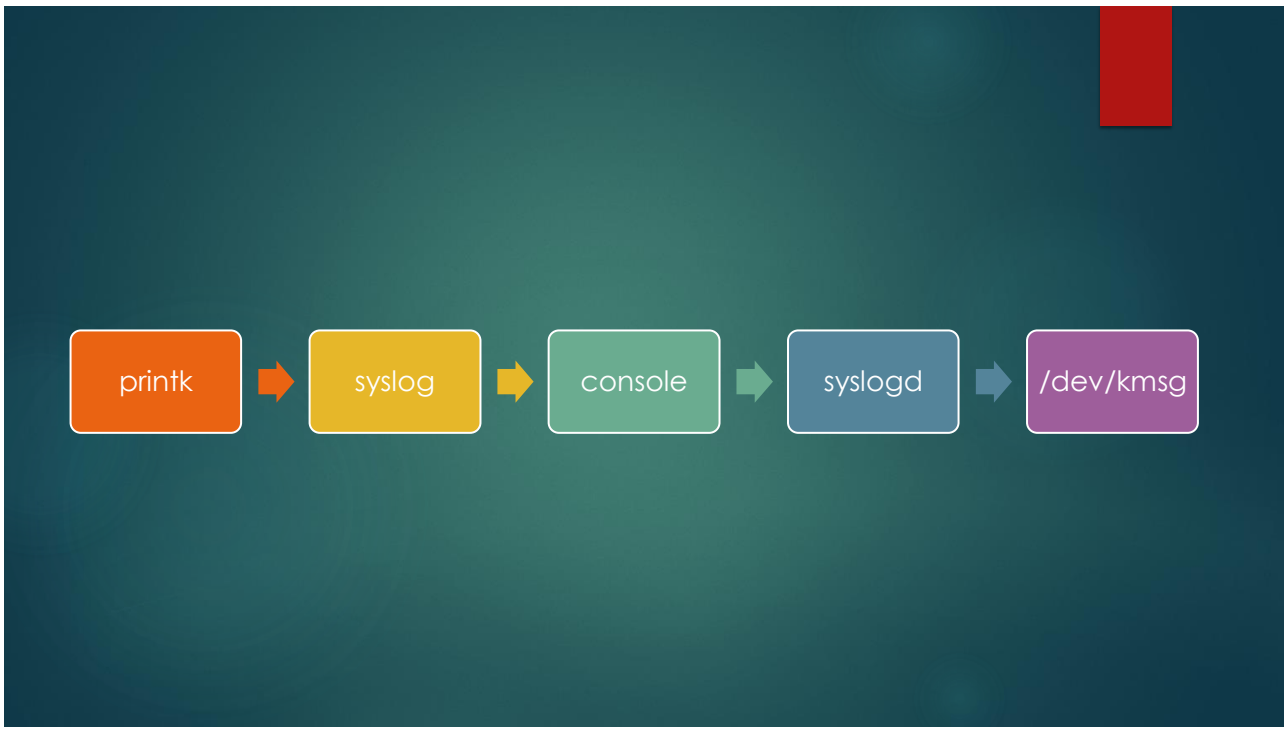
yk@yk: /var/log

```
Apr  2 03:24:10 yk systemd[1]: Started Wait for Network to be Configured.
Apr  2 03:24:10 yk systemd[1]: Starting Initial cloud-init job (metadata service crawler)...
Apr  2 03:24:10 yk cloud-init[965]: Cloud-init v. 19.4-33-gbb4131a2-0ubuntu1~18.04.1 running 'init' at Thu, 02 Apr 2020 03:24:00 +0000. Up 5959.62 seconds.
Apr  2 03:24:10 yk cloud-init[965]: ci-info: ++++++++++++++++++++++++++++++++++++++Net device info++++++++++++++++++++++++++++++++++++++
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +---------+------+-----------------------------+---------------+--------+-------------------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: | Device |  Up  |           Address           |      Mask     | Scope  |     Hw-Address    |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +---------+------+-----------------------------+---------------+--------+-------------------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |  eth0  | True |         10.10.0.5           | 255.255.255.0 | global | 00:0d:3a:c7:27:4e |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |  eth0  | True | fe80::20d:3aff:fec7:274e/64 |       .       |  link  | 00:0d:3a:c7:27:4e |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   lo   | True |         127.0.0.1           |   255.0.0.0   |  host  |         .         |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   lo   | True |          ::1/128            |       .       |  host  |         .         |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +---------+------+-----------------------------+---------------+--------+-------------------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: ++++++++++++++++++++++++++++++++Route IPv4 info++++++++++++++++++++++++++++++++
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +-------+-----------------+-----------+-----------------+-----------+-------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: | Route |   Destination   |  Gateway  |     Genmask     | Interface | Flags |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +-------+-----------------+-----------+-----------------+-----------+-------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   0   |     0.0.0.0     | 10.10.0.1 |     0.0.0.0     |    eth0   |   UG  |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   1   |    10.10.0.0    |  0.0.0.0  |  255.255.255.0  |    eth0   |   U   |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   2   |  168.63.129.16  | 10.10.0.1 | 255.255.255.255 |    eth0   |  UGH  |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   3   | 169.254.169.254 | 10.10.0.1 | 255.255.255.255 |    eth0   |  UGH  |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +-------+-----------------+-----------+-----------------+-----------+-------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: ++++++++++++++++++++Route IPv6 info++++++++++++++++++++
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +-------+-------------+---------+-----------+-------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: | Route | Destination | Gateway | Interface | Flags |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +-------+-------------+---------+-----------+-------+
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   1   |  fe80::/64  |    ::   |    eth0   |   U   |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   3   |    local    |    ::   |    eth0   |   U   |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: |   4   |   ff00::/8  |    ::   |    eth0   |   U   |
Apr  2 03:24:10 yk cloud-init[965]: ci-info: +-------+-------------+---------+-----------+-------+
Apr  2 03:24:10 yk ntfs-3g[1017]: Version 2017.3.23 integrated FUSE 28
Apr  2 03:24:10 yk ntfs-3g[1017]: Mounted /dev/sdb1 (Read-Only, label "Temporary Storage", NTFS 3.1)
Apr  2 03:24:10 yk ntfs-3g[1017]: Cmdline options: ro
Apr  2 03:24:10 yk ntfs-3g[1017]: Mount options: ro,allow_other,nonempty,relatime,fsname=/dev/sdb1,blkdev,blksize=4096
Apr  2 03:24:10 yk ntfs-3g[1017]: Ownership and permissions disabled, configuration type 7
Apr  2 03:24:10 yk ntfs-3g[1017]: Unmounting /dev/sdb1 (Temporary Storage)
Apr  2 03:24:10 yk systemd[1]: Reloading.
Apr  2 03:24:10 yk cloud-init[965]: Generating public/private rsa key pair.
Apr  2 03:24:10 yk cloud-init[965]: Your identification has been saved in /etc/ssh/ssh_host_rsa_key.
Apr  2 03:24:10 yk cloud-init[965]: Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub.
Apr  2 03:24:10 yk cloud-init[965]: The key fingerprint is:
```

50

```
Apr  2 03:24:10 yk systemd[1]: Reloading.
Apr  2 03:24:10 yk cloud-init[965]: Generating public/private rsa key pair.
Apr  2 03:24:10 yk cloud-init[965]: Your identification has been saved in /etc/ssh/ssh_host_rsa_key.
Apr  2 03:24:10 yk cloud-init[965]: Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub.
Apr  2 03:24:10 yk cloud-init[965]: The key fingerprint is:
Apr  2 03:24:10 yk cloud-init[965]: SHA256:96qSa72McgygrPUlwgZ641HD0h4f2pGCegccEeubpWM root@yk
Apr  2 03:24:10 yk cloud-init[965]: The key's randomart image is:
Apr  2 03:24:10 yk cloud-init[965]: +---[RSA 2048]----+
Apr  2 03:24:10 yk cloud-init[965]: |  oo             |
Apr  2 03:24:10 yk cloud-init[965]: |  ..             |
Apr  2 03:24:10 yk cloud-init[965]: | ..=   .         |
Apr  2 03:24:10 yk cloud-init[965]: | .B B +          |
Apr  2 03:24:10 yk cloud-init[965]: |.+.O.B oS .      |
Apr  2 03:24:10 yk cloud-init[965]: |+.*=O =  . .     |
Apr  2 03:24:10 yk cloud-init[965]: |.=E* = +    .    |
Apr  2 03:24:10 yk cloud-init[965]: |....o *o.  .     |
Apr  2 03:24:10 yk cloud-init[965]: |     +oo+o.      |
Apr  2 03:24:10 yk cloud-init[965]: +----[SHA256]-----+
Apr  2 03:24:10 yk cloud-init[965]: Generating public/private dsa key pair.
Apr  2 03:24:10 yk cloud-init[965]: Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Apr  2 03:24:10 yk cloud-init[965]: Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
Apr  2 03:24:10 yk cloud-init[965]: The key fingerprint is:
Apr  2 03:24:10 yk cloud-init[965]: SHA256:HHV+ipGs362h/AHueeft7o4NrN7xsx1qxoomC8mPUko root@yk
Apr  2 03:24:10 yk cloud-init[965]: The key's randomart image is:
Apr  2 03:24:10 yk cloud-init[965]: +---[DSA 1024]----+
Apr  2 03:24:10 yk cloud-init[965]: |          . .    |
Apr  2 03:24:10 yk cloud-init[965]: |         o +     |
Apr  2 03:24:10 yk cloud-init[965]: |        . + . .  |
Apr  2 03:24:10 yk cloud-init[965]: |       . o o o   |
Apr  2 03:24:10 yk cloud-init[965]: |        S o .    |
Apr  2 03:24:10 yk cloud-init[965]: |   E...  o o o   |
Apr  2 03:24:10 yk cloud-init[965]: |  . o+    o = =. |
Apr  2 03:24:10 yk cloud-init[965]: |   o  +. +.o.XoOo|
Apr  2 03:24:10 yk cloud-init[965]: |    .. o+.==O++*%|
Apr  2 03:24:10 yk cloud-init[965]: +----[SHA256]-----+
Apr  2 03:24:10 yk cloud-init[965]: Generating public/private ecdsa key pair.
Apr  2 03:24:10 yk cloud-init[965]: Your identification has been saved in /etc/ssh/ssh_host_ecdsa_key.
```

51



52

配置Serial Console

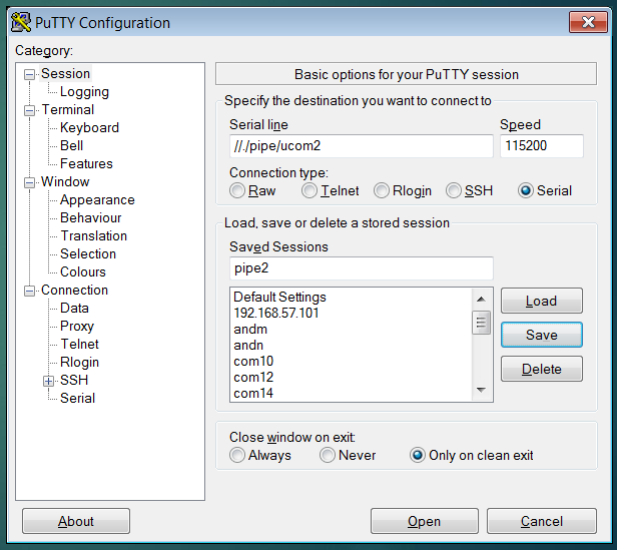
```
menuentry 'Ubuntu, with Linux 3.12.2 Serial Console on COM1' --class ubuntu --class gnu-linux --class gnu
--class os {
        recordfail
        gfxmode $linux_gfx_mode
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='(hd0,msdos1)'
        search --no-floppy --fs-uuid --set=root eb5d8aee-6a0f-4257-b5b8-8d6731ba6764
        echo 'Loading Linux 3.12.2 with serial console on COM1...'
        linux   /boot/vmlinuz-3.12.2 root=UUID=eb5d8aee-6a0f-4257-b5b8-8d6731ba6764 ro debug splash consol
e=ttyS1,115200n8
        echo 'Loading initial ramdisk ...'
        initrd  /boot/initrd.img-3.12.2
}
```

- /etc/grub.d/40_custom

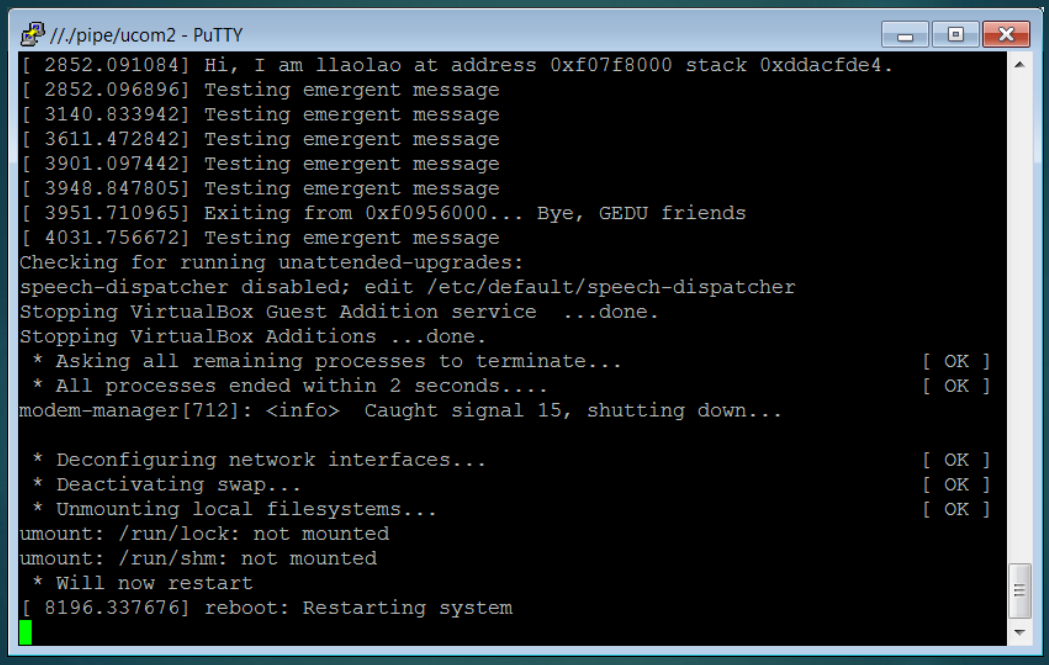
53

PuTTY

- PuTTY is a free (MIT-licensed) Win32 Telnet and SSH client.



54



55

消息屏蔽

```
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 */
};
```

- Messages below a predefined *loglevel* are displayed on the system console device.
- The default loglevel is defined in printk.c. In recent Linux kernels, it has the value 7

```
/*
 * Call the console drivers, asking them to write out
 * log_buf[start] to log_buf[end - 1].
 * The console_lock must be held.
 */
static void call_console_drivers(int level,
                                 const char *ext_text, size_t ext_len,
                                 const char *text, size_t len)
{
        struct console *con;

        trace_console(text, len);

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

56

cat /proc/sys/kernel/printk

```
4 4 1 7
```

- 控制台日志级别，级别更高（0-3）的才显示在控制台
- 默认的消息日志级别（ KERN_WARNING ），调用printk时没有指定日志级别时会使用这个值
- 控制台日记级别的允许最小值，dmesg –n 0会失败
- 默认的控制台日志级别，即第一项的默认值

```
/* We show everything that is MORE important than this.. */
#define MINIMUM_CONSOLE_LOGLEVEL 1 /* Minimum loglevel we let people use */
#define DEFAULT_CONSOLE_LOGLEVEL 7 /* anything MORE serious than KERN_DEBUG */
```

57

改变控制台日志级别

1. 1.内核启动参数Kernel boot option：loglevel=level
2. 2.通过dmesg命令: dmesg -n level
3. 3. echo $level > /proc/sys/kernel/printk
4. 4. 写程序使用syslog系统调用
5. 5. 修改sysctl.conf文件中的printk行

58

/etc/sysctl.conf

```
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3

##############################################################3
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
#net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguration
#  based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

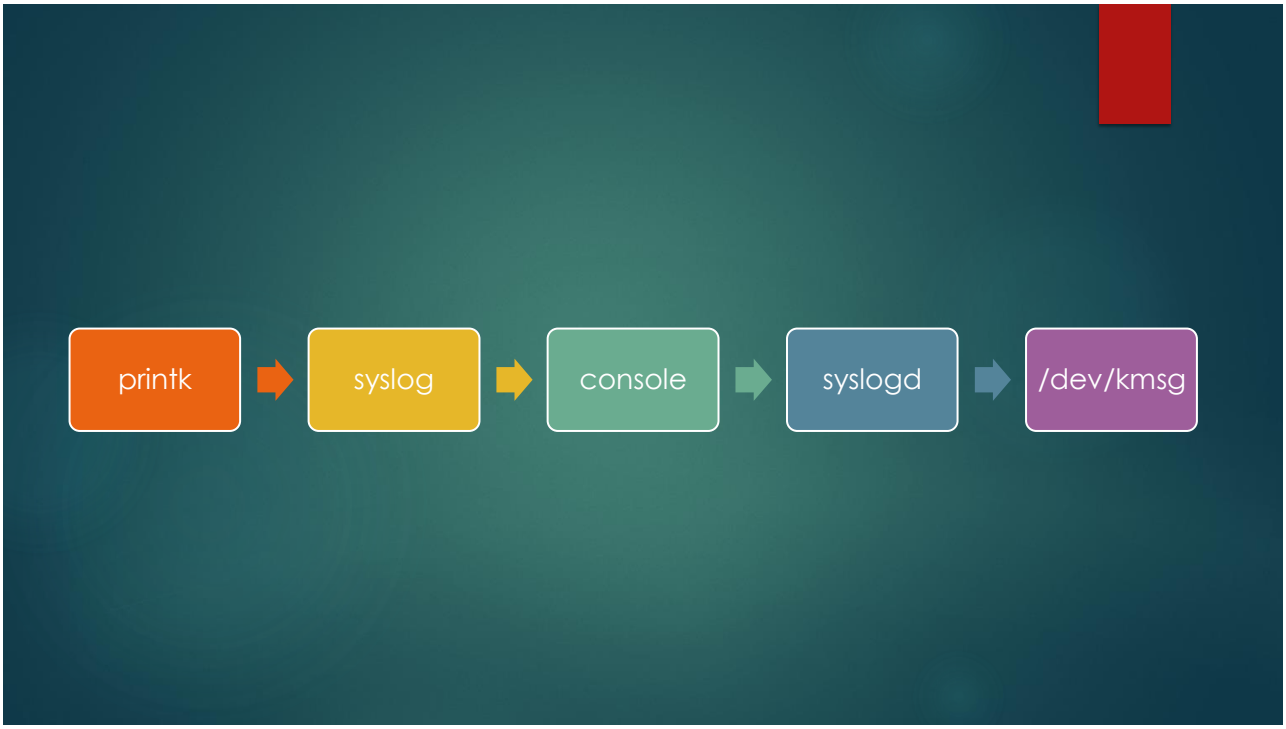
- # nano /etc/sysctl.conf
- kernel.printk = 5 4 1 7

59

Kernel command line options

debug— Sets the console loglevel to 10
quiet— Sets the console loglevel to 4
loglevel=— Sets the console loglevel to your choice of value

60



61

syslogd

```
ge@gewubox:~/work/linux-3.12.2$ ps -A | grep syslogd
  936 ?        00:00:00 syslogd
```

- 捕捉消息，根据规则写到文件
- 规则在文件*/etc/syslog.conf*

```
ge@gewubox:/etc$ ls /var/log
alternatives.log  dist-upgrade    hp              mail.warn          ufw.log
apt               dmesg           installer       messages           unattended-upgrades
auth.log          dmesg.0         jockey.log      news               upstart
boot              dmesg.1.gz      jockey.log.1    pm-powersave.log   user.log
boot.log          dmesg.2.gz      kern.log        samba              vboxadd-install.log
bootstrap.log     dmesg.3.gz      lastlog         speech-dispatcher  vboxadd-install-x11.log
btmp              dmesg.4.gz      lightdm         syslog             VBoxGuestAdditions.log
ConsoleKit        dpkg.log        lpr.log         syslog.1           VBoxGuestAdditions-uninstall.log
cups              faillog         mail.err        syslog.2.gz        wtmp
daemon.log        fontconfig.log  mail.info       syslog.3.gz        Xorg.0.log
debug             fsck            mail.log        udev               Xorg.0.log.old
```

62

规则语法

Facility. Severity **Action**

- mail.inf /var/log/maillog
  - append message from the mail facility with a level of info or higher to /var/log/maillog
- mail.=debug /ar/log/maillog.debug

main.c | syslog.conf

```
auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none      -/var/log/syslog
#cron.*              /var/log/cron.log
daemon.*             -/var/log/daemon.log
kern.*               -/var/log/kern.log
lpr.*                -/var/log/lpr.log
mail.*               -/var/log/mail.log
user.*               -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info            -/var/log/mail.info
mail.warning             -/var/log/mail.warn
mail.err             /var/log/mail.err

# Logging for INN news system
#
news.crit            /var/log/news/news.crit
news.err             /var/log/news/news.err
news.notice          -/var/log/news/news.notice
```

63

The facility - 产生消息的部件

- kern is the kernel message.
- user is the user-level message.
- mail is the mail system.
- daemon is the system daemon.
- auth is the security authorization message.
- syslog is messages generated internally by syslogd.
- lpr is the line printer.
- news is network news.
- uucp is the UUCP system.
- cron is the clock daemon.
- authpriv is the security/authorization message.
- ftp is the ftp daemon.

64

Severity

- emerg means that the system is unusable.
- alert means that action must be taken immediately.
- crit specifies a critical condition.
- err is an error condition.
- warning is a warning condition.
- notice is a normal but significant condition.
- info is informational.
- debug is the debug level.

65

Action

- 文件或者以下内容：
- Named pipe
- Console/terminal
- Remote host
- List of users
- All users currently logged on

66

文件循环使用

logrotate.conf

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}
```

67

/var/log

- /var/log/messages: native services, the kernel logger, the network manager, boot process, mail services, cron jobs, and many other services that do not have their own log files. can be considered to be a global log file of sorts
- /var/log/boot.log: the system boots
- /var/log/maillog: mail server
- /var/log/secure: authentication and authorization privileges
- /var/log/wtmp: user login records
- /var/log/btmp: failed login attempts
- /var/log/cron: cron (and anacron)(命令调度，执行预定任务)
- /var/log/lastlog: binary log that contains all of the last login information
- /var/log/yum.log: Yum and reports any activity related to the server's package management tools

68

观察日志文件

- # tail -n 100 /var/log/maillog
- # tail -f /var/log/maillog

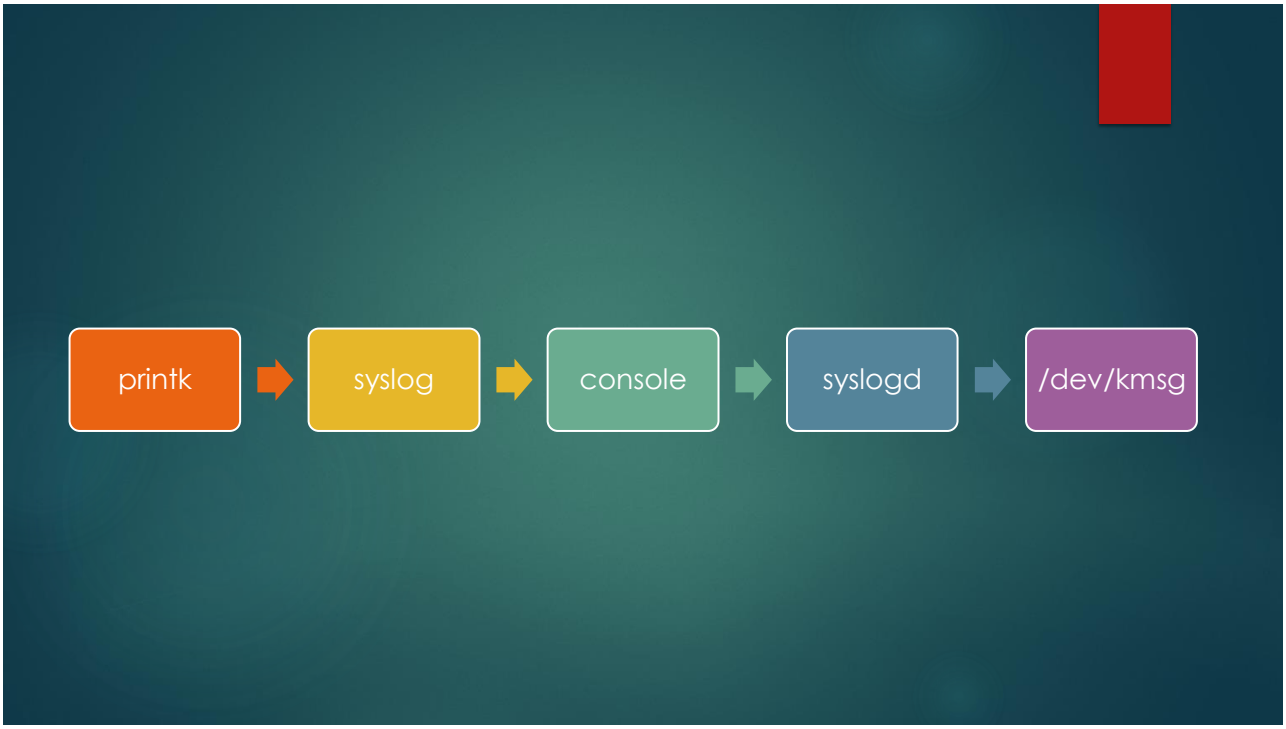
```
NAME
       tail - output the last part of files

SYNOPSIS
       tail [OPTION]... [FILE]...

DESCRIPTION
       Print  the  last  10 lines of each FILE to standard output.  With more than one FILE, precede each with a header giving the
       file name.  With no FILE, or when FILE is -, read standard input.

       Mandatory arguments to long options are mandatory for short options too.

       -c, --bytes=K
              output the last K bytes; alternatively, use -c +K to output bytes starting with the Kth of each file

       -f, --follow[={name|descriptor}]
              output appended data as the file grows; -f, --follow, and --follow=descriptor are equivalent

       -F     same as --follow=name --retry

       -n, --lines=K
              output the last K lines, instead of the last 10; or use -n +K to output lines starting with the Kth

       --max-unchanged-stats=N
              with --follow=name, reopen a FILE which has not changed size after N (default 5) iterations to see if  it  has  been
```

69

搜索

- # grep -R "XXX.XXX.XXX.XXX" /var/log/
  - 子目录
- # grep -Rn "XXX.XXX.XXX.XXX" /var/log/
  - 显示行号
- # grep -E "term 1 | term 2 | term 3" /var/log/messages
  - 搜索多个项

70



71

/dev/kmsg

- 虚拟文件接口
- Exports the structured data in the following line format:
  - "<level>,<sequnum>,<timestamp>,<contflag>[,additional_values, ...];<message text>\n"

```
const struct file_operations kmsg_fops = {
        .open = devkmsg_open,
        .read = devkmsg_read,
        .write_iter = devkmsg_write,
        .llseek = devkmsg_llseek,
        .poll = devkmsg_poll,
        .release = devkmsg_release,
};
```

72

dmesg

- 打印以及设置系统日志（环形缓冲区）
- 支持多种方式读取系统日志
  - /dev/kmsg
  - syslog()系统调用
  - 文件映射
- utils/util-linux/sys-utils/dmesg.c
- ftp://ftp.kernel.org/pub/linux/utils/util-linux/

```
569 /*
570  * Top level function to read messages
571  */
572 static ssize_t read_buffer(struct dmesg_control *ctl, char **buf)
573 {
574   ssize_t n = -1;
575
576   switch (ctl->method) {
577   case DMESG_METHOD_MMAP:
578           n = mmap_file_buffer(ctl, buf);
579           break;
580   case DMESG_METHOD_SYSLOG:
581           if (!ctl->bufsize)
582                   ctl->bufsize = get_syslog_buffer_size();
583
584           n = read_syslog_buffer(ctl, buf);
585           break;
586   case DMESG_METHOD_KMSG:
587           /*
588            * Since kernel 3.5.0
589            */
590           n = read_kmsg(ctl);
591           if (n == 0 && ctl->action == SYSLOG_ACTION_READ_CLEAR)
592                   n = klogctl(SYSLOG_ACTION_CLEAR, NULL, 0);
593           break;
594   }
595
596   return n;
597 }
```

73

```
NAME
       dmesg - print or control the kernel ring buffer

SYNOPSIS
       dmesg [options]

       dmesg --clear

       dmesg --read-clear [options]

       dmesg --console-level level

       dmesg --console-on

       dmesg --console-off

DESCRIPTION
       dmesg is used to examine or control the kernel ring buffer.

       The default action is to read all messages from kernel ring buffer.

OPTIONS
       The --clear, --read-clear, --console-on, --console-off and --console-level options are mutually exclusive.

       -C, --clear
              Clear the ring buffer.

       -c, --read-clear
              Clear the ring buffer contents after printing.

       -D, --console-off
              Disable printing messages to the console.

       -d, --show-delta
              Display the timestamp and time delta spent between messages. If used together with --notime then only the time delta
```

74

使用

- # dmesg -T | less
- # dmesg -T | grep -i memory
- dmesg | mail -s "Help me understand" goodfriend@example.com
  - Send the output of dmesg to a friend for advice, if you're really stuck.

75

Linus Torvalds suspends key Linux developer

Kernel panic as Systemd dev pokes the bear

RELATED

Software ▸ Developer

Torvalds rails at Linux developer: 'I'm f*cking tired of your code'

Kay Sievers banished to fuming Finn's doghouse

https://lkml.org/lkml/2014/4/2/420
http://www.theregister.co.uk/2014/04/05/torvalds_sievers_dust_up/

Date Wed, 2 Apr 2014 11:57:41 -0700
Subject Re: [RFC PATCH] cmdline: Hide "debug" from /proc/cmdline
From Linus Torvalds <>

On Wed, Apr 2, 2014 at 11:42 AM, Steven Rostedt <rostedt@goodmis.org> wrote:
>
> The response is:
>
> "Generic terms are generic, not the first user owns them."

And by "their" you mean Kay Sievers.

Key, I'm f*cking tired of the fact that you don't fix problems in the code *you* write, so that the kernel then has to work around the problems you cause.

Greg - just for your information, I will *not* be merging any code from Kay into the kernel until this constant pattern is fixed.

This has been going on for *years*, and doesn't seem to be getting any better. This is relevant to you because I have seen you talk about the kdbus patches, and this is a heads-up that you need to keep them separate from other work. Let distributions merge it as they need to and maybe we can merge it once it has been proven to be stable by whatever distro that was willing to play games with the developers.

But I'm not willing to merge something where the maintainer is known to not care about bugs and regressions and then forces people in other projects to fix their project. Because I am *not* willing to take patches from people who don't clean up after their problems, and don't admit that it's their problem to fix.

Kay - one more time: you caused the problem, you need to fix it. None of this "I can do whatever I want, others have to clean up after me" crap.

Linus

76

On Thu, Apr 03, 2014 at 08:17:33AM -0700, Tim Bird wrote:
>
> I had no idea systemd was so verbose and was abusing the kernel
> log buffers so badly. I'm not a big fan of the rate-limiting, as this just
> seems to encourage this kind of abuse.

That was a bug in systemd, and has been fixed up in the latest versions,
so it shouldn't happen anymore, even with debugging enabled.

thanks,

greg k-h
--

http://lkml.iu.edu/hypermail/linux/kernel/1404.0/01945.html

77

对调试设施的理解深度以及能否合理使用反应了一个软件工程师的成熟度

2016/8/21

78

切问而近思

欢迎关注格友公众号

79