

PB173 – Ovladače jádra – Linux III.

Jiří Slabý

ITI, Fakulta Informatiky

12. 10. 2010

LDD3 kap. 4 (zastaralá)

Jak zjistit chybu v jádře

- Ladicími výpisy
- Výstupem přes soubor (obsah velkých bufferů apod.)
- Pád ⇒ analýza Oops
- Regrese ⇒ `git bisect`
- Automatické nástroje
- (Debugger – `kgdb` a další)
- ...

- `printk` již známe
- Před formátovací řetězec se vkládá `KERN_*`
 - Řetězec ve formě `<číslo>`
 - Sděluje úroveň (důležitost) zprávy
 - Definované v `linux/kernel.h`

Důležitost	Konzole	<code>/var/log/*</code>	<code>dmesg</code>
vysoká	ANO	ANO	ANO
normální	NE	ANO	ANO
ladicí	NE	NE	ANO
zapisuje	jádro	<code>syslog*</code>	jádro

Obvyklé nastavení

Úroveň ovlivňuje, co se kde zobrazí

- `dmesg` nastavit nelze
- Nastavení `syslogu` závislé na systému
- Nastavení konzole
 - `/proc/sys/kernel/printk`
 - 4 čísla:
 - 1 *Zprávy s vyšší úrovní se objeví na konzoli* (`dmesg -n`)
 - 2 Zprávy bez úrovně dostanou tuto
 - 3 Minimum pro bod 1
 - 4 Nevyužito
 - Hodnota 8 \Rightarrow všechno dění
 - Např. při „hard-locku“

Demo: `pb173/03 + /proc/sys/kernel/printk + dmesg -n`

Volba úrovně s rozvahou

- Nechceme 10G `/var/log/messages` (DEBUG)
- Nechceme spoustu hlášek na konzoli (WARNING, ERR)

Co vše může být konzole pro výpisy

- `tty0-tty*` (co když stroj umře?)
- Sériová konzole (`ttyS*`)
- Síťová konzole (`netconsole`)
- Všechno, co se zaregistruje pomocí `register_console`

Nastavení pomocí parametru jádra (modulu)

- `console=tty1`
- `console=ttyS0`
- `netconsole=...`

[Documentation/console/console.txt](#)

[Documentation/serial-console.txt](#)

Použití netconsole

- 1 Na hostovi: `nc -ul 0.0.0.0 60000`
- 2 `dhclient`, je-li třeba
- 3 `dmesg -n 8` (vypisovat vše)
- 4 `modinfo netconsole`
- 5 `modprobe netconsole`
`netconsole=@/,60000@10.0.2.2/`
- 6 Např. `echo h >/proc/sysrq-trigger`

Více informací v [Documentation/networking/netconsole.txt](#)

- Mnoho dat
 - Znepřehlednění (typické pro ACPI, iwlmwifi)
 - Nestíhá se posílat na konzoli (ztráta informací)
 - *Řešení* (částečné): `if (printk_ratelimit()) printk();`
- Binární data
 - `dmesg` zobrazuje jen textová data
 - *Řešení* (částečné): `print_hex_dump_bytes (linux/kernel.h)`

Mnoho binárních dat se předává pomocí speciálních souborů

- Většinou v `/sys/kernel/debug/`
 - Pokud ne: `mount -t debugfs none /sys/kernel/debug/`
- `Documentation/filesystems/debugfs.txt`
- `linux/debugfs.h`
- `debugfs_create_dir,`
`debugfs_create_file (file_operations),`
`debugfs_create_symlink,`
`debugfs_create_u{8,16,32,bool},`
`debugfs_remove`
- Lze si předat data, jsou pak v `inode->i_private` v `open`

Vytvoření položek v debugfs

- 1 Vytvořit adresář
- 2 Vytvořit v něm soubor (u16)
 - Vrací např. 11320
- 3 Přeložit, vložit do systému
- 4 Přečíst soubor

- Chci-li znát komunikaci jádro ↔ program
- **man strace**
- Zachytává syscalls
- Vypisuje jejich parametry na STDOUT

Demo: `strace -e open,read cat .../debug/...`

- System-request (Alt-PrintScreen)
- Funguje (většinou) i po pádu systému
- Zapnutí: `sysctl kernel.sysrq=1`
- [Documentation/sysrq.txt](#)

h	zkrácená nápověda
r	„reset klávesnice”
p	výpis tras procesů na CPU
t	výpis tras všech procesů
w	výpis blokujících procesů
s-u-s-b	sync + unmount + reboot

Příklady SysRq

Pozn.: v Qemu lze vyzkoušet v ovládací konzoli (Ctrl-Alt-2) příkazem `sendkey alt-sysrq-p`

Analýza Oops

```
my_init(void) { char *ptr = (void *)5; *ptr = 5; }
```

BUG: unable to handle kernel NULL pointer dereference at 0000000000000005

IP: [<ffffffffffa0039042 >] my_init+0x12/0x30 [pb173]

PGD 3daff067 PUD 3c713067 PMD 0

Oops: 0002 [#1] SMP

last sysfs file: /sys/devices/system/...

CPU 0

Modules linked in: pb173(+) mperf fuse ...

Pid: 2506, comm: insmod Not tainted 2.6.36-rc4-16-default #1 /Bochs

RIP: 0010:[<ffffffffffa0039042 >] [<ffffffffffa0039042 >] my_init+0x12/0x30
[pb173]

RSP: 0018:ffff88003c741f28 EFLAGS: 00010292

RAX: 0000000000000017 RBX: ffffffff0039180 RCX: 0000000000000e91

...

CR2: 0000000000000005 CR3: 000000003bb47000 CR4: 00000000000006f0

Process insmod (pid: 2506, ...)

Stack: ...

Call Trace:

[<ffffffffff810002da >] do_one_initcall+0x3a/0x170

[<ffffffffff8108f69a >] sys_init_module+0xba/0x210

[<ffffffffff81002efb >] system_call_fastpath+0x16/0x1b

[<00007fc3386320ba >] 0x7fc3386320ba

Code: ... e8 63 c8 41 e1 <c6> 04 25 05 00 00 ...