

PB173 – Ovladače jádra – Linux

I.

Jiří Slabý

ITI, Fakulta Informatiky

27. 9. 2011

- Semestr = 12 týdnů (25. 10. Kernel Summit)
- Cvičení
 - Vývoj jádra od r. 2005 (NetBSD, Linux)
 - Student FI
- Cíle cvičení
 - Nastínit trochu jiný model programování
 - Prohloubit znalosti vnitřností OS a HW
- Ukončení: k
 - Splnění domácích úkolů
 - 2 body na úkol, alespoň $\frac{3}{5}$ z celkového počtu
- Vše potřebné ve studijních materiálech v ISu

- satyr01-satyr10
 - CentOS 5.x
 - Login/heslo: vyvoj/vyvoj
- COMBO6X karty
 - Později; k PCI, I/O, na přerušení, DMA
 - <http://www.liberouter.org/>

- GIT
 - Domácí úkol
 - Podrobněji: <http://book.git-scm.com/>
- Zdroje jádra
 - Použijeme předinstalované z RPM (`/usr/src/kernels/`)
 - <http://git.kernel.org/> (od 2.6.12) **Owned**
 - `full-history-linux` GIT (od 0.01 do 2.6.26)
- QEMU nebo jiná VM
 - Ze začátku

Práce s GITem

- 1 Domácí úkol
 - Dotazy?
- 2 Proveďte klon
 - `git clone git://github.com/jirislaby/pb173.git`
- 3 Prozkoumejte
 - Příklady ze cvičení
 - Adresář pro domácí úkoly

Výstup GITu = patch (záplata)

Záplaty v linuxovém jádře

- 1 Poslat odpovídajícímu správci (`scripts/get_maintainer.pl`)
 - 2 Poslat na ML „pull request” a vystavit celý GIT strom
- Popis v [Documentation/SubmittingPatches](#)

Stejným způsobem odevzdávání domácích úkolů

- Záplatováním se opravuje i přidává nová funkcionality
- Někdo odpovědný aplikuje záplatu/provede pull
 - Co když není správce opravovaného kódu?
- L. Torvalds má právo veta

Část I

Rozdíly

Hlavní rozdíly

- Žádné *libc* (`printf`, `strlen`, `malloc`, ...), ani ostatní (`pthread`)
- Ne/oddělený *paměťový prostor*
- *Počáteční funkce* (`main`)
- Pád systému \Rightarrow pád všeho

- GNU C (x86 už jen gcc \geq 4.x)
- *CodingStyle*
 - Kontrola: `scripts/checkpatch.pl` (ne 100%)
- Monolit (`vmlinux` \rightarrow (b) `zImage`)
 - Ale moduly (standardní ELF: `*.ko`)
- `module_init` (=main), `module_exit` (=on_exit)

Průzkum modulu (.ko objektu)

- 1 Zvolte si jeden modul v systému (`lsmod`)
- 2 Zavolejte na něm `modinfo`
- 3 Proveďte `objdump -d` sekce `.modinfo` sekce
- 4 Porovnejte oba výstupy

- `Documentation/*`
- Generovaná tamtéž (podobná DocBook)
- Kód (<http://lxr.linux.no/>)

Demo: pomocí lxr najděte v jádře obecnou (v `lib/`) a optimalizovanou (v `arch/`) implementaci `strlen`

Hello World

- 1 Spustte virtuální stroj
- 2 `git clone git://github.com/jirislaby/pb173.git`
- 3 Prozkoumejte adresář 01 z pb173 git repozitáře
 - `Makefile`, `pb173.c`
- 4 Do `init` funkce doplňte výpis „Hello World”
 - `printk(KERN_INFO "...");`
 - Objeví se v `/proc/kmsg` a na konzoli
- 5 Přeložte a vložte do systému
 - `make`, `insmod`
- 6 Zkontrolujte výstup `dmesg`
- 7 Můžete udělat `commit` a `push` do svého repozitáře

- Omezený zásobník (4k-8k)
 - Žádná nebo malá rekurze
 - Jen pro malá data (`int`, malé `struct`, krátká pole, ...)
- Pracuje se se stránkami (na x86 4K-1G)
 - Omezená velikost alokace (fragmentace)
 - Podrobnosti v dalších cvičeních

Malé alokace (sta bajtů, max. až cca. 4M)

- Horní mez závislá na architektuře
- `kmalloc`, `kfree`

Alokace mají (většinou) GFP parametr. Ten určuje, co si alokátor může dovolit (spát, swapovat, použít HIGHMEM, ...). Prozatím nám stačí `GFP_KERNEL`.

```
void *mem = kmalloc(100, GFP_KERNEL);  
if (mem) {  
    ...  
}  
kfree(mem);
```

- 1 V `exit` funkci naalokujte 1000 bytů
- 2 Udělejte do nich `strcpy` „Bye”
- 3 Vypište paměť jako řetězec
- 4 Uvolněte paměť
- 5 Vložte a odeberte modul ze systému
- 6 Můžete `commit+push`