

# PB173 – Binární programování Linux

## VIII. C bez libc

Jiri Slaby

Fakulta informatiky  
Masarykova univerzita

19. 11. 2015

- libc se stará o komunikaci s OS (jádreem)
  - Jádro poskytuje pouze „základní“ funkcionalitu
- libc poskytuje spoustu nadstavieb
  - Soubory: `fopen`, `fread`, `fwrite`, `fclose`, ...
  - Síť: `getaddrinfo`, `ntoh*`, `hton*`
  - Více vláken (`libpthread`): `pthread_*` (včetně zamykání)
  - A spoustu dalšího (celý POSIX), ...

**Bez libc nic z toho nemáme**

Jak teď ale např. otevřu soubor?

- Sdílená paměť
  - Jádro zapisuje, uživatel čte a naopak
  - V jednom z dalších cvičení
- Systémová volání
  - Podobné jako zavolání funkce
  - Ale program a jádro běží v jiných kontextech
  - Přepnutí kontextu
    - Uložení a změna stavu procesoru
    - Relativně drahá operace
  - Uvidíme dnes dále

- Volání funkce
- Instrukcí procesoru
  - Závislé na architektuře
  - Softwarové přerušení (x86\_32: číslo 0x80)
  - Speciální instrukce (x86\_32: `sysenter`, x86\_64: `syscall`)
  - **Demo:** implementace `syscall` z `glibc`
- Systemová volání jsou očíslovaná
  - Do jednoho registru číslo
  - Čísla `__NR_*` definovaná jádrem (pro každou architekturu)
  - `sys/syscall.h`
  - Např. (x86\_64): `__NR_write (1)`, `__NR_exit (60)`

## libc o úroveň níže

- 1 Zavolejte systémová volání
  - `write` s nějakým textem
  - `exit`
- 2 Proveďte ale pomocí funkce `syscall` z `libc`
  - Viz `man 2 syscall`
  - První argument: číslo systémového volání
  - Další argumenty: odpovídají už danému volání
  - Např. `syscall(__NR_kill, -1, SIGKILL)`
- 3 Přeložte a vyzkoušejte

- Žádné hlavičkové soubory z libc, ani jeho funkce
- Jen to, co poskytuje jádro (/usr/include/linux/)
- Při překladu: `gcc -nostdinc ...`
- Při linkování: `gcc -nostdlib ...`
  - `gcc` stále může vkládat volání `memset` apod.

## Volací konvence

Parametr	Registry			
	Systémová volání		Uživatelský prostor	
	i386	x86_64	i386	x86_64
1.	EBX	RDI	EAX	RDI
2.	ECX	RSI	EDX	RSI
3.	EDX	RDX	Zásobník	RDX
4.	ESI	R10	Zásobník	RCX
5.	EDI	R8	Zásobník	R8
6.	EBP	R9	Zásobník	R9
7.	<i>Jen 6 parametrů</i>		Zásobník	Zásobník
8.			Zásobník	Zásobník
Návratová	EAX	RAX	EAX	RAX

Pozn.: i386 a -mregparm

## Systemová volání bez `libc`

- 1 Otevřete a projděte si `pb173-bin/07/`
- 2 Zavolejte
  - `fork`
  - Z potomka: `write` na standardní výstup
  - Z rodiče: `read` 16 bytů a jejich `write` na standardní výstup
  - Z obou potom: `exit`
- 3 Přeložte a spusťte
  - `gcc -nostdlib ...`



- Jádro předává
  - Parametry programu
  - Proměnné prostředí
  - Rozšiřující vektor
- Vše na zásobníku
  - Formát zásobníku je pevně daný

# Formát zásobníku

## Formát zásobníku

Počet parametrů	<code>long</code>
1. parametr	<code>char *</code>
...	<code>char *</code>
m-tý parametr	<code>char *</code>
Konec parametrů	<code>OUL</code>
1. proměnná prostředí	<code>char *</code>
...	<code>char *</code>
n-tá proměnná prostředí	<code>char *</code>
Konec proměnných prostředí	<code>OUL</code>
Rozšiřující vektor 1	<code>struct { long type, val; }</code>
...	<code>struct { long type, val; }</code>
Rozšiřující vektor o	<code>struct { long type, val; }</code>
Konec rozšiřujících vektorů	<code>{AT_NULL, ? }</code>

## Typ záznamů

```
struct {  
    long type;  
    long value;  
};
```

- Typy: AT\_NULL, AT\_ENTRY, AT\_RANDOM, ...
- Pole struktur zakončuje typ AT\_NULL

## Zjištění názvu programu a platformy

- 1 Vypište první parametr programu
  - Tj. název samotného programu
- 2 Dále iterujte přes zásobník až k rozšiřujícímu vektoru
- 3 Tam najděte typ `AT_PLATFORM`
- 4 Vypište hodnotu jako řetězec
- 5 Přeložte (bez `libc`) a spusťte

## vsyscall

- Jen na některých architekturách
- Neprobíhá přepnutí kontextu
- Speciální stránka(y) s kódem namapovaným jádrem
- Podpora jen 3 funkcí
  - `gettimeofday`: aktuální čas (0xfffffffffff600000)
  - `time`: čas v sekundách od Epochy (0xfffffffffff600400)
  - `getcpu`: číslo CPU a NUMA uzlu (0xfffffffffff600800)
- **Demo:** `/proc/self/maps`

## Volání `time` z `vsyscall`

- 1 Zavolejte adresu s `time`
- 2 Vypište hodnotu jako řetězec
- 3 Přeložte a spusťte několikrát

## vdso

- Speciální knihovna přilinkovaná jádrem
- Podpora také závislá na architektuře
- Podobná `vsyscall`, ale více flexibilní
  - Navíc obsahuje ale jen `clock_gettime`
- **Demo:** `ldd` a `readelf`
- Knihovna v ELF formátu
  - Adresa začátku v `auxv`: `getauxval(AT_SYSINFO_EHDR)`
  - Dále se přečte ELF pomocí `libelf`
  - Ukázkový kód: `Documentation/vDSO/parse_vdso.c`

**Úkol:** domácí úkol