PB173 – Binární programování Linux

VII. C bez libc

Jiri Slaby

ITI, Fakulta informatiky

5. 11. 2013

C a `libc`

- `libc` se stará o komunikaci s OS (jádrem)
  - Jádro poskytuje pouze „základní” funkcionalitu
- `libc` poskytuje spoustu nadstaveb
  - 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 např. ale otevřu soubor?

Komunikace s jádrem

- 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
- Sdílená paměť
  - Jádro zapisuje, uživatel čte a naopak
  - V jednom z dalších cvičení

Systémová volání

- 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`
- Systémová 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)

Úkol

`libc` o úroveň níže

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

Virtuální systémová volání

`vsyscall`

- Jen na některých architekturách
- Neprobíhá přepnutí kontextu
- Speciální stránka(y) s kódem namapovaným jádrem
- Podpora jen 3 funkcí
  - `getcpu`: číslo CPU a NUMA uzlu
  - `gettimeofday`: aktuální čas
  - `time`: čas v sekundách od Epochy
- Demo: `ldd`
- Adresa funkce se získá pomocí makra `VSYSCALL_ADDR()`
  - `asm/vsyscall.h`
  - Parametr je název shora s předponou `__NR_v`

Úkol

Volání `time` z `vsyscall`

1. Zjistěte si adresu `time` z `vsyscall`
2. Zavolejte tento `time`
3. Vypište hodnotu
4. Přeložte a spusťte několikrát

C bez `libc`

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

Úkol

Systémová volání bez `libc`

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

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	Zásobník
7.	Jen 6 parametrů		Zásobník	R9
8.	–		Zásobník	Zásobník
Návratová	EAX	RAX	EAX	

Tabulka : Volací konvence

Pozn.: i386 a `-mregparm`

Virtuální dynamická knihovna

`vdso`

- Speciální knihovna přilinkovaná jádrem
- Podpora také závislá na architektuře
- Podobná `vsyscall`, ale více flexibilní
  - Navíc obsahuje 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