Posudek oponenta diplomové práce

Práce: Linux kernel userspace modules
Autor: Lukáš Lipavský
Vedoucí: Martin Děcký
Oponent: Petr Tůma

Práce navrhuje a implementuje podporu userspace modulů v kernelu systému Linux. Klíčovou částí řešení je napojení userspace modulu na kernel, to je dosaženo rozšířením kernelu o interpretr imperativního jazyka, ve kterém je zapsán kód volající userspace modul.

Analytická část práce je poměrně krátká a trpí nepřesným vyjadřováním, například když se v kapitole 2.1.2 tvrdí, že (žádný) purpose-specific interface nedovoluje přistupovat ke kernel internals a že nevyžaduje nový kód v kernelu (což se dá vyložit mnoha způsoby), nebo když v kapitole 3.2 uvádí, že je možné navrhnout interpretovaný jazyk tak blízký jazyku C, jak jen je to možné. I kapitoly 4.2 a 4.3, ve kterých se popisuje bytecode a interpretr, jsou napůl rozkročené mezi analýzou a referencí. Ze závažnějších otázek vybírám:

- Neměla by se analýza zabývat tím, jaké funkce (v širším chápání slova) může userspace modul od kernelu požadovat (volání funkcí, poskytování callback funkcí, přístup k datovým strukturám, přístup k hardware ...), aby pak mohla navrhnout odpovídající řešení ? Jak velké je v tomto kontextu například omezení, že userspace modul nemůže iniciovat komunikaci s kernelem ?

- Čím jsou motivována omezení interpretovaného jazyka ? Řada z nich je zdá se zvolena pouze kvůli jednoduchosti překladače a interpretru, nedávalo by větší smysl mít omezení zvolená vzhledem k použití jazyka ? Z detailních otázek k jazyku pak například není snadné nalézt odpovědi na tyto:

  - Co je any_type a proč je pointer na něj volně převáděn na void pointer a zpět ?
  - Proč není implementováno odčítání pointerů ?
  - Proč nejsou podporovány pointery na funkce ?
  - Proč musí mít různé definice stejné funkce uvnitř různých konstrukcí podmíněného překladu stejnou signaturu ?
  - Proč nebyly signatury funkcí navrženy tak, aby bylo možné sdílet headers mezi userspace modulem a interpretovaným kódem ?

- Jak velké omezení z hlediska userspace modulu je, že nejdou číst konstanty definované v kernelu ? Nebyl by lepší nějaký obecnější mechanismus než ručně napsané funkce get_HZ a get_PAGE_SIZE, podobně jako se to nakonec udělalo u inline funkcí ?

- Jak je to s rychlostí interpretovaného kódu ? Zde nemám ani tak na mysli absolutní rychlost, u které jistě platí uvedené úvahy o možnostech interpretru ve srovnání s překladačem. Zajímavější by asi bylo vědět, jaké typy úloh je možné userspace modulem implementovat bez zásadní ztráty výkonu.

- Jak se bude interpretovaná část modulu ladit ?

Pokud odhlédnu od zmíněných nedostatků analytické části práce, vidím nabízené řešení jako elegantní a po technické stránce rozhodně netriviální (osobně si nemyslím, že by v prostředí systému Linux bylo možné navrhnout nějaké zásadně lepší řešení, vždy bude nutná celá řada kompromisů). Věřím, že autor práce svým řešením jasně demonstroval schopnosti odpovídající absolventovi magisterského studia a proto práci doporučuji k obhajobě.

Ke kódu samotnému poznamenám (vedle obvyklého povzdechu nad malým objemem komentářů) ještě to, že bych očekával mírně složitější příklad, který by lépe prověřil použitelnost a stabilitu řešení. U stávajícího kódu v podstatě nemohu říci, že bych testoval o mnoho více než překlad.

V Praze 16. září 2008.

Petr Tůma