Využití koprocesoru Tesla pro interaktivní tvorbu a ladění programů v operačním systému SOS-4

Jan KRUPIČKA, Michael RUBÍN; Orgaprojekt, Praha

1. Úvodem - interaktivně nebo dávkově?

Ať se nám to líbí nebo ne, ladění programů patří k činnostem, jimiž se při své práci programátor zabývá nejčastěji. Tuto činnost může provádět dvěma způsoby:

A. Dávkově
B. Interaktivně

Zatímco interaktivní způsob ladění programů - díky zavádění terminálů - postupně převládl a stal se neodmyslitelnou součástí života programátora, neomezených možností dávkového ladění se v současné době většina programátorů přeochotně vzdává.

Při pozornějším zkoumání obou způsobů ladění však zjistíme, že oba jsou založeny na opakovaném provádění následující sekvence:

- Příprava zdrojového textu
- Kompilace
- Vyhledávání a identifikace chyb
- Oprava zdrojového textu
- Kompilace
  ... atd.

Ve fázi odstraňování syntaktických chyb se tato sekvence může nesčetněkrát opakovat, čímž programátor nakonec získá více či méně zdařilý program.

Po odstranění syntaktických chyb zpravidla následuje likvidace chyb logických, čehož dosáhneme několikerým opakováním poněkud upravené sekvence:

...
- Kompilace, zlinkování a případná katalogizace fáze
- Zkušební výpočet
- Vyhledávání a identifikace chyb
- Oprava zdrojového textu
- Kompilace, zlinkování a případná katalogizace fáze
- Zkušební výpočet

... atd.

Při srovnání obou variant zjistíme, že interaktivní ladění se od dávkového liší toliko způsobem manipulace se zdrojovým textem. Programátor ladící programy interaktivně mění zdrojový text pomocí editoru, při dávkovém ladění provádí změny přímo v děrných štítcích /nebo na disketě/, případně využívá možností programu "LIBR".

Možnosti měnit sled jednotlivých kroků v závislosti na výsledcích kompilace nebo výpočtu se nemusí příliš lišit. Vycházejme ze situace, že i při dávkovém ladění programů má programátor možnost dovědět se výsledek svého snažení krátce po skončení každého kroku. V obou případech programátor potřebuje k odhalení chyb protokol o překladu, případně sestavu ze zkušebního výpočtu.

2. Interaktivně = efektivně?

V operačním systému DOS-4 interaktivní překladače bohužel nejsou příliš rozšířeny a nevyskytují se ani specializované editory, které by programátora mohly upozornit na mnohé syntaktické chyby při manipulaci se zdrojovým textem. Určitou výjimku tvoří PASCAL, u něhož však není příliš velká naděje, že by se v dohledné době prosadil jako vhodný jazyk pro zpracování dat.

Bez protokolu o překladu se tedy v naprosté většině případů nelze obejít. Při ladění nám slouží k odhalování chyb, a protože výše uvedené sekvence mohou být mnohokrát opakovány, je programátor obvykle nucen jej tisknout tolikrát, kolikrát program překládá. Na konci snažení programátora je tedy vedle odladěného programu i hromada makulatury, přímo úměrná délce programu a počtu neúspěšných pokusů o odstranění chyb. Protokol o úspěšném pokusu se zpravidla uschovává pro dokumentační či jiné účely, ostatní programátor obvykle s klidným svědomím zahazuje.

Různá nepopulární opatření pro snížení spotřeby tabelačního papíru ovšem programátorovi značně komplikují život. Není tedy divu, že vynakládá často neuvěřitelného úsilí na to, aby dosáhl jejich úplného zrušení nebo alespoň nějaké výjimky pro svou osobu, svůj úkol, ... atd.

Když je pak najednou zásoba papíru ve výpočetním středisku s konečnou platností vyčerpána a /díky neutěšené situaci na našem trhu/ její obnovení je v nedohlednu, nebo jsou dokonce vyřazeny z provozu tiskárny, problém nabývá hrozivých rozměrů. V takovýchto situacích se programátoři uchylují k zoufalým /a často i velice bizardním/ pokusům obejít se bez tištěného protokolu. Opomineme-li různá zvěrstva typu "ASSGN SYSLST,SYSCON", stalo se nejužívanější pomůckou prohlížení protokolu přímo v LST-frontě systémem "LUISA". Toto napohled celkem vyhovující řešení s sebou ale nese mnohá nebezpečí.

"LUISA" totiž neumožňuje jednoduché střídání zdrojového textu a protokolu z LST-fronty na obrazovce terminálu a fáze "LUISW" /pro nemožnost svého využití v procedurách/ málokomu přirostla k srdci. Programátora, který k tomuto účelu využívá systému "LUISA", můžeme přirovnat k člověku, který si prohlédne protokol o překladu /výpočtu/, odloží jej a zpaměti se pokouší opravovat zdrojový text. Pochopitelně se mu málokdy podaří zapamatovat si, co že to všechno vlastně chtěl opravovat, a nezbývá mu tedy nic jiného, než se vrátit k již odloženému protokolu a znovu jej prohlížet. Pozornost takového člověka se tříští a jeho výkonnost

prudce klesá. Jestliže programátor musí několikrát opakovat již zmíněnou sekvenci kompilací a oprav, v LST-frontě se začínají hromadit zdánlivě stejné položky.

Použijeme nyní přirovnání k člověku, co neustále odkládá protokol, aby se k němu znovu vracel, a jehož situace se ještě komplikuje soustavným prohledáváním různých papírů a vyhledáváním právě aktuálního výpisu. Tento nešťastník se dříve či později nutně ocitá na pokraji zoufalství, kdy se, obklopen nepřehlednou změtí papíru, marně snaží zachovat si alespoň zbytek lidské důstojnosti. Produktivita jeho práce klesá k nule, škod, spáchané na našich lesích jsou evidentní.

Toto napohled absurdní přirovnání zajisté většina programátorů odmítne, vyhledat poslední položku v LST-frontě není přece problém, ale přidají-li se ještě různé vnější vlivy, můžeme se stát svědky marných pokusů opravovat chyby v položkách LST-fronty a ještě mnohem smutnějších výjevů.

Interaktivní ladění, které až do této chvíle bylo těžkou záležitostí, se stává nesnesitelným utrpením, programátor, soustavně vystavený stresům, postupně chátrá a ztrácí radost ze života. Nepodlehne-li zcela těmto nepříznivým okolnostem, pozbývá zcela své profesionální hrdosti i dalších, pro život mnohem důležitějších vlastností.

3. Minieditor "M2D" - prostředek k záchraně

V době, kdy jsme se postupně začali smiřovat s osudovou neměnností tohoto stavu, objevil se v operačním systému DOS-4 nenápadný minieditor "M2D", nikdy předtím se o něm nemluvilo, jeho zavedení se obešlo bez humbuku a nadšeného vychvalování, v celé systémové dokumentaci je mu věnována jediná kapitola v rozsahu dvaceti stran. Jeho výstup je většinou směrován na SYSIOH, v dokumentaci se píše o jakýchsi opravných štítcích pro "LIBR", formát obrazovky málokomu něco připomínal, navíc bylo nutno pro jeho používání umět funkční klávesy terminálu a "SLIBA", koneckonců,

není tak úplně nejhorší. Snad právě proto se jeho používání příliš nerozšířilo, ale postupně upadal v zapomenutí.

Při jeho bližším zkoumání jsme však zjistili, že minieditor "MED" má řadu velice příjemných vlastností, které jsme u systému "LUISA" zcela postrádali. Jeho obsluha sice nitím nepřipomíná systém "LUISA", zato je velice jednoduchá /jenom dokumentace pro systém "LUISA" zabírá zhruba desetkrát více stránek/ a kupodivu snadno zapamatovatelná /autoři tohoto příspěvku se minieditor "MED" naučili efektivně využívat přibližně za šest hodin v průběhu tří dnů/. Kromě snadné obsluhy se minieditor "MED" vyznačuje i možností současného zpracování několika souborů najednou, jejich snadného střídání na obrazovce bez vyloučení možnosti dávkového zadávání parametrů.

Minieditor "MED" obsahuje prakticky všechny funkce potřebné pro opravování zdrojových programů, je ideálním prostředkem pro použití v různých procedurách a nechybí mu ani tak potřebná uživatelská přítulnost.

Minieditor "MED" tedy úspěšně řeší problém oprav zdrojového textu při současném prohlížení protokolu v LST-frontě. Stačí jen navrhnout a sestavit nerůznější ladicí procedury a o naše programátory se dále nemusíme obávat.

Tato cesta se však záhy ukázala jako značně neschůdná, neboť při navrhování procedur pro ladění programů se nám nepodařilo vyhnout se zadávání příkazů "MGO"/"MGOB"/, ani jsme uspokojivě nevyřešili případ výskytu více zdánlivě stejných položek v LST-frontě. K tomu přibyla nutnost vytvořit nejméně dvě procedury pro každý programovací jazyk /jednu pro opravu a překlad programu, druhou pro opravu s prohlížením protokolu/, které se navíc různě parametrizovaly /vytvoření dočasné nebo trvalé fáze, katalogizace modulu/.

Kdo někdy zažil něco podobného, zajisté nám dá zapravdu v tom, že ani tato cesta není optimální. Vždyť jenom pouhý překlep v příkaze "GO" /ve stavu "PAUSE"/ vede k neočekávaným a mnohdy zcela nepochopitelným zvratům v průběhu dalšího zpracování. Navíc je nutno

v tomto případě dvakrát mačkat "ENTER".

Vytvořením procedur pro minieditor "MED" jsme se tedy zastavili v půli cesty a přínos našich snah byl přinejmenším sporný. Programátor by tak sice dostal mnohem přítulnější prostředek, než byla dosud "LUISA", ale vše ostatní by zůstalo při starém. Navíc by se musel učit novým způsobům práce s dosud neznámou komponentou operačního systému, osvojovat si nové znalosti s dost pochybným účinkem. Tak by snadno mohla nastat situace, kdy každá komplikace /kterých se i v ladění pomocí systému "LUISA" může vyskytnout celá řada/ byla automaticky kladena za vinu šílenému nápadu s tím ještě šílenějším nesmyslem /rozuměj: minieditorem "MED"/. Z programátora, který byl ještě včera přístupný rozumným argumentům, se pak může rázem stát zatvrzelec, nad nímž neustále visí reálná hrozba naprosté degradace, a pro něhož není návratu.

4. Jde to i bez procedur

Není-li možné běžnou komunikaci s výpočetním systémem uspokojivě zautomatizovat pomocí procedur, můžeme se pokusit využít služeb makroprocesoru TAMu.

Jeho možnosti jsou bez nadsázky srovnatelné s možnostmi operátora terminálu. Operátor se ovšem může splést, nebo nemusí být vždy ochoten dělat za programátora rutinní práci. Makroprocesor TAMu je však připraven sloužit vždy. Jeho programy /tak zvaná terminálová makra/ sice nikoho neobtěžují chybami E008, E017, ... atd., zato však nesprávně navržené terminálové makro dokáže v rekordně krátkém čase zcela zdecimovat i nejotrlejšího uživatele terminálu.

V našem V. S. jsme v minulosti získali nějaké zkušenosti s terminálovými makry, a tak jsme se celkem bez obav odhodlali k jejich nasazení i zde, při využívání minieditoru "MED". Naší ideou bylo vytvoření ladícího systému, který by většinu rutinních a často se opakujících úkonů vykonával za programátora, a jehož ovládání by bylo co nejjednodušší.

Tento systém by měl zůstat i po své aktivaci zcela skrytý, měl by v postavení bezprávného otroka čekat na případný pokyn programátora, aby jej mohl přesně a spolehlivě vyplnit. Tak vznikl ladicí systém "SERVUS".

Prvním problémem, který měl tento systém vyřešit, se stalo zjednodušení ladění programů pomocí dosud existujících procedur. Bylo-li nutné rozhodnout /po dokončení překladu ... atd./, v proceduře se provedl příkaz "PAUSE", na který "SERVUS" reagoval vyslá-
ním dotazu k programátorovi. Jeho odpověď pak vyhodnotil a na jejím základě vydal příslušný příkaz "MGO" nebo "MGOB", aniž se kdy dopustil překlepu nebo zapomněl na klávesu "ENTER".

Postupně se ukázalo, že tento způsob komunikace je dosti těžkopádný, a proto jsme vytvořili sadu nových procedur, tentokrát určenou pouze pro využití v systému "SERVUS". Tyto procedury už neobsahovaly žádný příkaz "PAUSE", ale reprezentovaly nejčastěji používané sekvence příkazů /vyvolání editoru "MED", uložení opraveného textu do knihovny, vyvolání překladače, test RSC-kódu/. Jelikož jsme usilovali o použitelnost systému "SERVUS" pro jakýkoli překladač dodávaný s operačním systémem, museli jsme zvyšovat počet parametrů jednotlivých procedur, až se situace stala neúnosnou. Rozšíření repertoáru procedur by vedlo ke vzniku nepřehledného monstra, a tak jsme se opět rozhodli pro terminálová makra. Pro většinu funkcí jsme vytvořili samostatná terminálová makra, která jsou aktivována hlavním makrem. Tak jsme postupně eliminovali téměř všechny procedury a ponechali jsme pouze ty, které se nejvíce osvědčily.

5. Univerzální ladicí systém

V současné době systém "SERVUS" nabyl jakési standardní podoby, která se pravděpodobně už nebude příliš měnit.

Po zadání příkazu "BATCH ... EMPTY" a po založení oddílu programátor asynchronním příkazem aktivuje "SERVUS", který se ohlásí, ale programátora žádným způsobem neobtěžuje ani neomezuje. Komunikuje prostřednictvím zadaného čísla oddílu a programátorovi se jeví jako rozšíření množiny příkazů MONITORu. Stejným způsobem tedy indikuje zadání chybného příkazu a umožňuje jejich opravu. V případě nejasností se dotazuje na další postup a na případné chyby programátora jemně upozorní.

"SERVUS" umožňuje ladění programů psaných pro překladače ASSEMBLERU, COBOLU, FORTRANU - E, FORTRANU - H, PASCALU a PL/1 - - opt. U prvních dvou navíc využívá "OPTION DECK", takže po úspěšném překladu se programátor může rozhodnout, zda chce vytvářet dočasnou nebo trvalou fázi, aniž by bylo potřeba znovu program překládat. Protokoly o překladu zůstávají uloženy v LST-frontě a "SERVUS" si udržuje přehled o tom, který z nich je aktuální. Ani spouštění minieditoru "MED" pro současné opravy zdrojového textu a prohlížení protokolu nemusí programátorovi dělat ty nejmenší starosti.

Mimo ladění programů v běžných programovacích jazycích "SERVUS" nabízí možnost ladění terminálových i assemblerských maker, ladění programů v PLS, výpisy adresářů knihoven a disků, vyvolání samotného minieditoru "MED" nebo systému "LUISA".

"SERVUS" důsledně dodržuje koncepci vyloučení tiskáren z běžné práce programátora, a proto bylo nutno pro některé výpisové funkce /adresáře knihoven/ vytvořit speciální programy, pro některé bylo možno využít systémových programů /stupeň zaplnění knihovních souborů/.

Kromě toho ladicí systém "SERVUS" obsahuje i důležitou funkci "HELP", která neznalému programátorovi poskytne vyčerpávající informace o jednotlivých funkcích i o celém systému "SERVUS".

6. Závěrem

Tímto příspěvkem jsme se pokusili upozornit na nepříliš známé možnosti operačního systému DOS-4 a na nové funkce, které nám nabízí spojení zdánlivě nesouvisejících komponent operačního systému.

Kromě toho případ ladicího systému "SERVUS" ilustruje pracovní postup, kdy během řešení problému byly postupně upřesňovány i požadavky uživatelů, nevyhovující řešení byla zavrhována a několikrát bylo nutno začít téměř od začátku.

Výsledkem tohoto našeho snažení je téměř universální ladicí systém, který programátorovi všemožně ulehčuje jeho neradostný úděl a umožnuje mu zvýšit produktivitu práce. U programátorů, kteří běžně používají ladicí systém "SERVUS", se dále prokazatelně snížila spotřeba tabelačního papíru téměř o 50 % a jsme přesvědčeni, že totéž je možno tvrdit i o hladině adrenalinu v jejich krvi.

Ladicí systém "SERVUS" je předmětem zlepšovacího návrhu číslo 14/88, jehož správcem je ORGAPROJEKT Praha.