MOVE - IN - UP TELEKOMUNIKAČNÍ SYSTÉM PRO TVORBU INTERAKTIVNÍCH ÚLOH

Ing. Václav Maňásek, PS Ústí n/Labem

Proč byl prostředek vytvořen:

V našem oddělení jsme vždy inklinovali k interaktivnímu zpracování, jak při ladění, tak v orientaci uživatelských úloh.
V prvním bodě jsou /dle mého soudu/ systémové prostředky naprosto vyhovující /především zásluhou IUISY/.
V bodě druhém je stav daleko horší.
Postupně jsme zkoušeli /samozřejmě na ostrých agendách, nikoli na zkušebních datech předem naaranžovaných tak, aby nečinily potíže/ snad všechny dosažitelné komponenty.
Nejvíce jsme se zabývali SMLV, zkoušeli jsme však i DUORS, COBOLské DISPLAY, UPON terminal atd. Nejdříve jsme vyloučili prostředky, které neumožňují full-screen. Tyto jdou použít pouze ve specielních případech a nemají obecný charakter. Prostředky ostatní nás udivily svou pracností, rozsáhlostí, omezeními a malou účinností.
Proto jsem začal vymýšlet a posléze vyvíjet prostředek vlastní, který při minimálních znalostech programátora a při co nejmenší pracnosti produkuje účinný, snadno ovladatelný aparát.
/Jinými slovy: aby i průměrný programátor neznalý Assembleru a tím méně maker Supervisoru mohl běžně používat všech možností a výhod makra TAMMSG full-screenového formátu/.

Obecné charakteristiky :

Při konstrukci kladu důraz na to, abych postupoval zdola nahoru tak, aby komponenty již hotové mohly být samostatně používány. /Vyšší usnadňuje nižší/.
Dle mého soudu je naprostý nesmysl začít např. telekomunikačním monitorem, když nemá na co navázat.
Konečný produkt by /snad/ měl mít tyto základní úrovně:

1. tvorba obrazovek v programu
2. editor obrazovek
3. telekomunikační Supervisor umožňující práci s mnoha terminály v jedné úloze
4. telekomunikační Monitor.

Zatím se mi do slušné míry dokonalosti povedlo přivést pouze 1.bod.

Editor bude hotov co nejdříve. Nevím, jestli Supervisor má vůbec význam, neboť většina podniků nedisponuje desítkami terminálů, nicméně už chodí. Jak bude vypadat Monitor, vůbec netuším.
Ve výkladu se podrobněji zmíním o bodu 1., tj. konstrukce obrazovek v programu a jejich využití. Nastíním však i body 2., 3.

Specifické charakteristiky

Prostředek je vázán na COBOL DOS-4. Dotváří jej jakoby o další - celoobrazovkovou - periferii. Je možný zápis na obrazovku a čtení určených položek z obrazovky. Jedna obrazovka tvoří asi tolik co větu. Nemá však žádný ID ani nic podobného.

Délka položek je omezena pouze COBOLem a délkou obrazovky. /Zde odbočím: soudím, že pro obecný případ je chybné omezovat délku alfa-numerické položky. Bude-li např. obrazovka typu text, je ji nutno popsat a číst jako jedinou položku a tu teprve nějak dekódovat/.

Je možno zobrazovat i číst běžné COBOL položky. Zobrazení i čtení se děje dle COBOL v popisu. Numerické položky jsou tedy editovány a při čtení konvertovány dle PIC. Je samočinně kontrolována jejich přípustnost a případně požadována oprava.

Položkám i celým úsekům lze udělovat různé světelné atributy, obrazovku lze rastrovat, řádkovat, existují i prostředky pro snadnou tvorbu pohyblivých obrázků. Lze jednoduše zjistit adresu kurzoru, dekódovat PF-klíče, číst bez odmáčknutí ENTER nebo PF, vrátit původní stav obrazovky klávesou CLEAR. Lze inicializovat položky jednotlivě nebo na celé obrazovce apod./ viz popis/.

Technika programování :

Základní úroveň MOVE - IN - UP tvoří 2 moduly:

PALL - vytváří formát obrazovky, v jeho USING se uvádí popis obrazovky tvořený :
- COBOL položkami a alfanum.literály /texty/
- operátorem řádku /A/N literál ' - '/
- světelnými atributy /dekadický literál 0 - 9/
- umístěním na obrazovce /dekadický literál se znaménkem/

MALL - zobrazí formát a případně čte /při čtení vždy platí WRIN/. V jeho USING se uvádí seznam čtených položek, případně se

jako 1. parametr může použít název paragrafu, který obsahuje /pouze/ formátování. /Takto se napomáhá strukturovanosti programu/. Neuvedeme-li žádný parametr, pouze píše a nečeká na načtení /pohyblivé obrázky apod./.

Možno uvést ještě modul TALL. Tento je ve všem shodný s MALL až na to, že čte ENTRu nebo PF. /Možno použít k samočinnému listování, ale i k tvorbě her apod./.

Strategie programování poněkud připomíná práci s DISPLAY UPON CONSOLE a ACCEPT FROM CONSOLE WRIN.
Nejdříve se obrazovka naformátuje PALLem, pak se MALLem zobrazí a případně načte. Nemění-li se formát, není třeba PALL znovu volat. To je vše, jednodušší to snad nemůže být.

Systém pracuje s terminálem, který je zároveň konzolou úlohy. Tento je samočinně přepnut /žádný ASSGN, TCC apod./ do stavu full-screen a pak vrácen do LINE. Tento systém se nám jeví /na rozdíl od všeobecného mínění/ naprosto nejvýhodnější.

Příklad :

```
CALL  'PALL' USING                                    1
'_  '  '_  '                                          2
+15   4 'MOVE = IN = UP '                             3
+16  POL                                              4
CALL  'MALL 'USING POL                                5
```

Vysvětlení po řádcích :

1. volání PALL pro formát
2. vynech 2 řádky
3. na 17. pozici 3. řádku piš dvojnásobně jasný text /4/ uvedený alfanum. literálem
4. pod tímto textem zobraz položku POL /před tím někde deklarovanou/, dle jejího popisu.
   Dejme tomu, že položka POL stojí ve WORKING-STORAGE a má tento popis :
   Ø1 POL PIC S 999 V 99 COMP - 3 VALUE 123.45

Obrazovka pak vypadá takto:

```
MOVE IN UP
+ 123.45
```

Numerická položka je samočinně editována. Kurzor je umístěn pod ní, očekává se její načtení, neboť je uvedena v MALL. /ř.5/ S komplikovaným zápisem /např./ maker SMLV nelze tento způsob vůbec srovnávat.

Editory :

Je vyvíjen editor obrazovek TELE a jeho nadstavba ZIPS. Jsou plně interaktivní /aktivitu má více méně editor/. TELE pracuje podobně jako editory mikropočítačů. Z popisu obrazovky generuje volání modulů s příslušnými řetězci a vše ukládá pod zadaným jménem do SLB. Je možno zadávat texty, jména položek, atributy atd. Popisy v SLB se dají lehce aktualizovat. Úsek se pak volá příkazem COPy.

Supervisor DEE - GEE

Tuto komponentu zatím ještě nelze použít. Pracuje jako multitásk, i když nevenek se tak nejeví. /Pro zajímavost uvedu,že se těžko hledá makro SPV, které nebylo použito/.
Základní myšlenka telekomunikačního Supervisoru je takováto : Program pracuje s několika terminály, které se připojují LOGON /na rozdíl od PAL-MALL/. Program by však měl být konstruován běžně tak, jakoby pracoval pouze s jedním terminálem /žádné ID terminálu apod./. Jakýkoliv terminál se kdykoliv správně připojí a správně připojí a správně pokračuje /jde v COBOLu stejným úsekem, ale jiným subtáskem, jakoby v jiné dimenzi/. Netřeba uvádět, že programování tohoto Supervisoru je problematické.

Příklady již zaběhnutého použití PALL-MALL:
Aktualizace datové základny /ISAMU/. Sběr dat z formulářů.
Procedurální přehledy, sumarizace a výběry archivů dat.
Komplexní systém aktualizace matice PSV.

Místo konečného zhodnocení konstatuji toto :
PVT Praha si u našeho podniku objednal /a již obdržel/ 25 obrazovek do agendy BIPL. Nikoli tedy opačně, my u PVT, ale PVT u nás.

Problém účinného editoru obrazovek /editor kontra needitor/

Není těžké vytvořit editor, který by produkoval obrazovky typu "formulář". Poněkud složitější je zavést do takovéto obrazovky položku jménem, jaké má v příslušném zdrojovém textu. Většina

editorů /např. u DUORSu, ale i u mikropočítačů/ umožňuje pouze vytvořit statickou kostru z textů a do ní /někdy i značně kostrbatě/ napasovat položky deklarované v programu /dvojí deklarace atd./
Editor MOVE - IN - UP, tj. TELE, pracuje v několika režimech.
V režimu položkovém se názvy položek píší přímo na své místo na obrazovce, tedy žádné dodatečné užití screen-adresy, formátu na obrazovce atd. TELE zatím nemá spojení se zdrojem a tak formát položky se neukáže automaticky při konstrukci obrazovky, ale až při běhu programu. Zavést světelný atribut není těžké, záleží pouze na eleganci, s jakou to bude provedeno. Dle mého soudu je bezpodmínečně nutné, aby se světelný úsek zobrazil již při tvorbě obrazovky /nikoli tedy např. znak # znamená dvojnásobný jas a ukáže se až v chodu programu/. Existují editory /např. mikropočítačů/,které do absurdna parcelují jednu obrazovku a výsledkem je opět jenom formulář. Takováto obrazovka se dá celkem běžně udělat editorem jednodušším, bližším režimu "text".
Řekl bych, že tento stav je zapříčiněn nedostatečným rozšířením a odzkoušením terminálových aplikací. Alespoň kdykoli jsem se s kýmkoliv o tomto bavil, řekl, že mu to takto vyhovuje, neboť si zřejmě nic jiného nedokázal představit. Nuže není tomu tak /jak praví klasik nadrealismu/. Představme si např. sekvenční prohlížení souboru tak, že každá věta tvoří řádek terminálu. Jsou požadovány pohyby po řádcích, obrazovkách, vpřed vzad. Na vyžádání se ukáží určené sumy. Hlavička obrazovky se mění dle změny klíče /třeba střediska/. PALL-MALL toto umožňuje celkem jednoduše naprogramovat,ale editor běžného typu je jenom na obtíž. Ještě jakž takž by se dal napasovat při pouhém prohlížení. Ale co když budeme chtít téměř všechny položky obrazovky /z různých vět/ aktualizovat? Označit je jako akceptované by také nějak šlo. Ale při konverzi zpět je práce editorem jenom ztížena. Pochopitelně se také dá zodpovědně a zasvěceně prohlásit, že od tohoto editor není. Ale co když je? Pro uvedený příklad klasický soubor /i ISAM!/ více méně nevyhovuje, je stvořen pro READ WRITE po větách /není nepodstatné, že jenom dopředu/ pro full-screenový režim vlastně tento přístup definován není /větou je jakkoliv sestavená obrazovka/. Soubor by nejspíše měl být virtuální, po úsecích tvořící obrazovky se pohybujeme indexem.
A tak v zápětí vyskakuje další problém :
Mějme na obrazovce 100 indexovaných položek virtuálního souboru /což je běžné/. Tady je editor rovněž v koncích, muselo by se hezky

postupně všechny položky vypsat.
PALL-MALL tuto situaci řeší pomocí položek typu CBS. Tyto simulují WRITE /i READ/ po položkách na obrazovku, tato se pak zobrazí i čte najednou. Editor však stojí opodál. /Teoreticky vzato/ může se stát, že máme dost času a chceme naprogramovat hru. Není problém udělat statické obrázky. Co však s těmi pohyblivými? Stačí třeba jenom zkusit udělat, že chumelí. /Rozumí se jednoduše - pomocí editoru/.
Tak tedy: kde končí povinnost a pravomoc editoru a kde začíná povinnost a pravomoc překladače? Má editor umět udělat jiný editor /třeba souboru/? Nu, existují systémy, které bez editoru za moc nestojí, kdežto s editorem také ne /tak to řekl o něčem jiné Werich, ale hodí se to na všelicos/. Zatím to docela stačí, přeji si, aby doopravdy jenom zatím.

Komponenty, které přímo nesouvisí s tvorbou obrazovek, jsou však pro interaktivní zpracování důležité.

Sdílení isamských souborů v COBOLu DOS-4

V našem VS je systém dosti úzce svázán s datovou základnou uloženou jako ISAMY. /Prohlížení, aktualizace, ale i procedurální akce jako sumarizace atd./.
Ať už je soubor sdílený nebo nesdílený /tj. RESERVE ALTIME nebo DYNAMIC/, není ho možné běžnými prostředky zpracovávat na několika terminálech najednou /i třeba stejnými programy/, neboť :

1/ Je-li nesdílný, je situace jasná. Je-li soubor již otevřen, pak se již žádný další OPEN neprovede a čeká se / a to beze zprávy, což mi připadá trochu neslušné - rozumí se od makra OPEN/. Takto lze tedy se souborem pracovat pouze od jednoho terminálu, ale pozor, program blokuje jakýkoliv jiný program s tímto souborem, což může mít za následek velmi nepříjemná nedorozumění /např. v provozem/.
Takovýto soubor je tedy vhodný pro sobecké povahy.

2/ Soubor je sdílný /má RESERVE DYNAMIC, návod uveden např. v učebnici COBOLu u instrukce UFO/.
Takto lze zpracovávat pouze soubory klasicky /nikoli přes terminál/, neboť načtení věty blokuje celý soubor /nikoli tedy

pouze větu/.
Běžný postup aktualizace ISAMU přes terminál - tedy :

```
READ ISAM
CALL 'PALL'USING položky isamu
CALL 'MALL'USING aktualiz.položky
      ---aktualizace---
      ---řádově až minuty---soubor blokován---
REWRITE VETA-ISAMU
```

nelze použít, neboť soubor je po READU blokován, tj. druhý program stejně nejede a čeká.
Uvedeme-li po READU instrukci WRITE NO nebo PUT NO - pro odblokování /rovněž uvedeno u COBOLu/ REWRITE abnormálně skončí na špatnou sekvenci maker.

Při přímém přístupu by šel tento postup obejít. Kupodivu jsme však zjistili, že nejžádanějším přístupem k souboru je START na klíč a pak sekvenční přístup. /Nikoliv tedy, jak by se zdálo čistý RANDOM/.

Není těžké domyslet, že tomuto postupu je dosažitelný aparát vyloženě proti srsti.
Sekvenční přístup z několika terminálů k několika větám najednou je též nemožné /použití PUT NO nedovolí REWRITE, použití REWRITE, který rovněž odblokuje, přechází na novou větu/.
Situaci řeším takto:
Před OPENem ISAMu je třeba zavolat:

```
CALL'EVERY'USING ISAM
```

Toto volání nahodí sdílený soubor /který se zatím řídí všemi pravidly o zablokování/.
Po READu je soubor třeba odblokovat pro použití jiným programem:

```
CALL 'FREE' USING ISAM
```

Modul FREE se diametrálně liší od maker WRITE NO a PUT NO, neboť po sobě povoluje REWRITE. Po něm může se souborem pracovat jakýkoliv jiný program, aniž by narušil sekvenci předešlého. Je možno užít tedy běžného postupu pouze s jedním CALLem navíc.
Modul FREE toho povoluje až moc a tak je tedy možné, aby 2 programy načetly stejnou větu. Tato nežádoucí situace bude vyřešena.
Pozn.: Moduly provádějí své akce přepsáním DTF ISAMU. Toto nijak neovlivní žádné další akce.