VÝPOČETNÍ SYSTÉMY PŘÍŠTÍCH LET A JEJICH DOPAD NA PROFESNÍ SPĚCH

Ing. Zdeněk Rusín, Vítkovice, k.p., Správa operačních systémů, Ostrava 6, psč. 706 02

Článek se pokouší o charakteristiku netriviálního výpočetního systému budoucího desetiletí. Reálným vzorem je operační systém VME/B vyvinutý firmou ICL pro počítačovou řadu 2900, provozovaný od roku 1978 ve VS VŽaKG na počítači ICL 2960.
Úplnost tohoto operačního systému a samozřejmost těch jeho funkcí, jejichž řešení v jiných u nás provozovaných systémech mnohdy absorbuje nemalou programátorskou kapacitu, přesvědčují o účelnosti jeho koncepce, byť by se našla řada výtek k detailům implementace.

1. Výchozí koncept výpočetního systému

Základními všeobecně aplikovatelnými východisky našeho popisu budou tyto skutečnosti:
stavebnicová struktura hardware, umožňující zaměnitelnost hw prvků jejich generačními následovníky,
existence komunikačních sítí,
dynamické rekonfigurování komunikací, paměťových bloků i ostatního hardware,
likvidace děrných medií na vstupu a výstupu,
databázové struktury dat s uživatelskými obrazovkovými aplikacemi v reálném čase, vyznačující si prostředí virtuálního stroje v interaktivních úlohách,
omezování objemných tiskových výstupů a dávkového zpracování.

Z hardwarového hlediska připojme předpoklad výpočetního systému jako sítě mikroprogramovaných účelových procesorů s centrálním postavením procesorů řídícího kódu a paralelními asynchronně fungujícími řídícími procesory přístupu k paměti a periferiím, včetně řízení diskových souborů obsahem či kontextem věty. Zde mikroprocesory tvoří další stupeň sprostředkovanosti mezi uživatelem a fyzickým systémem. Závěrem náš koncept doplňme požadavky vysokého uživatelského komfortu, bezpečnosti živých dat uživatelů, jednotnosti interakcí mezi všemi typy uživatelů a operačním systémem a samozřejmostí strukturovaného projektu oper. systému.

2. Pojem centrálního katalogu

tak jako může stavebnicový prvek hardware počítače patřit k různým generacím své řady,může být jediný procesor řízen mikroprogramy různé dokonalosti a komplexnosti. Alespoň některé kombinace hardware a firmware musí být realisovány jako víceméně ekvivalentní konfigurační stavy počítače v rámci jediné verse operačního systému. To si vyžaduje apriorně automatickou evidenci fysických zdrojů systému nutnou pro informaci při oživení systému. Táž evidence je žádoucí či dokonce nezbytná v oblasti software, počínaje stavem operačního systému samotného,pro diagnostiku chyb software a tím spíše nutná v oblasti živých uživatelských dat včetně uživatelského programového vybavení. Tam navíc vždy jde mimo evidenci dat i o jejich účinnou ochranu před zneužitím i před přepsáním medií.

Všechny tyto požadavky lze řešit centrálním systémovým katalogem zahrnujícím popis všech objektů,jimiž je výpočetní systém tvořen fysicky nebo jež musí postihovat logicky. Takto pojatý katalog objektů hardware,software a uživatelské sféry,obsahující popisy vlastností objektů,vztahy nadřízenosti a podřízenosti, vlastnictví a práv přístupu jedněch objektů - vlastníků a uživatelů - k jiným objektům,se stává základním prvkem koncepce systému,koncepce stejně otevřené možnostem popisu reálné skutečnosti pomocí katalogového uzlu katalogovaného objektu,jako je fysická stavebnice systému otevřená možnostem vývoje i extensivního rozšiřování hardware.

3. Hierarchie procesů a pojem virtuálního stroje

Popisovanou flexibilitu hardware,logicky zajišťovanou aktualisací katalogu,respektuje hierarchická struktura procesů sdílejících společně reálný čas výpočetního systému. Inicialisační procedura oživení operačního systému přerůstá v rodičovský proces několika dalších autonomních procesů zajišťujících různé systémové funkce a řídících procesů různých typů počítačového zpracování,jež se stávají hierarchickými vlastníky koncových procesů uživatelských. Z hlediska vnějšího pozorovatele jsou tyto procesy rozčleněny do tzv. virtuálních strojů /dále jen VM - virtual machine/.

Exekutiva operačního systému,tj. mechanismus stránkování virtuálních pamětí jednotlivých VM,mechanismus přidělování procesorů a paměti procesům,procedury obsluhy a aktualisace katalogu, procedury inicialisace VM,funkce interakcí s periferiemi a další činnosti plynoucí z uspořádání operačního systému,tvoří prvý VM. Druhý v pořadí je VM operátoských interakcí hlavního operátorského stanoviště. Tyto dva VM koexistují společně prakticky po celou dobu chodu systémových procesů,přičemž přítomnost operátorského VM je podmíněna jen obvyklým požadavkem manuální ovládatelnosti systému. Podle pokynů operátora,na jehož stanovišti archaický dálnopis nahradily obrazovky s klávesnicí,jsou inicialisovány řídící procesy jednotlivých typů uživatelských úloh /dávkové zpracování, obrazovkový servis,transakční zpracování v reálném čase/.
Z uživatelského hlediska jsou tyto řídící procesy zdánlivě cizorodým znepřehledňujícím prvkem. Uživatele totiž zpravidla zajímá koncový proces jeho úlohy,realisovaný v jedinečném VM,podřízeném příslušnému řídícímu procesu,přes který vzniká a zaniká,který určuje jeho základní charakteristiky,který ale není zprostředkovatelem mezi uživatelskými programy a funkcemi operačního systému.

Hierarchie procesů se zvnějšku jeví jako pouhý paralelismus virtuálních strojů,modifikovaný stupnicemi priorit plánovacího mechanismu paměti a procesorů.

4. Některé důležitější procesy

Kromě výše zmíněných činností prvého virtuálního stroje a řídících procesů uživatelského zpracování existují další nezbytné procesy,např. mechanismus registrace chyb hardwaru magnetických medií,proces připojování uživatelských obrazovkových stanic pro práci v interaktivním servisu,jež nemusí být trvale katalogovány, proces účtování spotřeby zdrojů systému uživatelům a další.
Zvláštní kategorií koncových procesů jsou úlohy vstupu děrných medií /zpravidla pouze štítků s popisy dávkových úloh/ a výstupu tiskových a grafických souborů,které se důsledně realisují mimo uživatelskou úlohu tzv. SPOOLING systémem. Operační systém přebírá i funkci plánování výstupů,pochopitelně s možností operátorských zásahů. Spooling implikuje existenci fronty požadavků na výstupy s příslušnými funkcemi operátorských akcí změny pořadí atd. Tímtéž způsobem je organisován vstup úloh do několika front a ob-

dobně funguje i fronta komunikace s operátorem.

S výjimkou úloh transakčního zpracování jsou ostatní VM koncových uživatelských procesů reusabilní. VM hostí v čase postupně několik úloh jistého stupně příbuznosti. Chyba v koncové úloze likviduje v konečné instanci pouze proces této úlohy,nikoli samotný VM. Čím nákladnější je inicialisace VM,tím výhodnější je toto uspořádání.

5. Dokumentace o činnosti procesů

Vzhledem ke koexistenci mnoha uživatelských procesů a dalších VM jiné povahy,je nutné zavést zcela jiný způsob dokumentace chodu operačního systému i jednotlivých úloh. Uživatelskou úlohu doprovází deník úlohy,pro systémové činnosti existuje několik deníků globálních /účtování a výkonnostní údaje,přehled chodu úloh, záznam incidentů software apod./. Součástí deníku úlohy jsou statistiky výkonnostních údajů /celkový čas,čas procesoru,počet instrukcí,sumace zdrojových nároků/ a pochopitelně protokol o činnosti úlohy. Lokální uživatelské zprávy nezatěžují operátora. Odtud ovšem plyne změna koncepce dávkové úlohy - úloha musí s minimální potřebou operátorských zásahů automatisovaně řešit všechny své stavy. Toto umožňuje řídící jazyk systému.

6. Řídící jazyk systému

Na místo děrných štítků popisu úlohy přichází bohatě rozvinutý jazyk jednotný pro všechny typy uživatelů /operátor,systémový manažer,správce databázových systémů,programátor,problémový uživatel/. Je to blokově strukturovaný jazyk blízký Pascalu,umožňující vytváření řídících programů úloh různé složitosti,od primitivního sledu interpretovaných výkonných příkazů v obrazovkové úloze po kompilovanou proceduru zabezpečující všechny alternativní a chybové stavy dávkové úlohy provozního charakteru.

Přesto,že vzhledem k různým uživatelům plní různé funkce, vzhledem k operačnímu systému plní tento jazyk funkci jedinou - parametrisaci uživatelských požadavků a volání uživatelských programů a systémových procedur /funkcí/. Tato úloha spolu s jednoduchou syntaxí jej předurčují i na místo pseudokódu ve fázi projekční činnosti. Jazyk řízení systému obsahuje všechny jazykové konstrukce strukturovaného programování. Byť z hlediska uživatelského jde o nárokování a uvolňování zdrojů /media,paměť,procesory/,

každá systémová funkce má charakter manipulace s katalogovými objekty a jejich vlastnostmi. Jazyk proto obsahuje příkazy operací se znakovými řetězy a bitovými maskami využívané při manipulacích se jmény a vlastnostmi katalogových objektů.

Na počítači existuje strojově orientovaný assemblerový jazyk, zdrojové texty procedur operačního systému jsou však převážně generovány ve vyšším systémovém jazyku. V nejlepším případě je pak jazyk řízení systému jeho podmnožinou.

7. Hierarchické uspořádání operačního systému

Operační systém je vybudován hierarchicky. Kolem inicializačního, stránkovacího a synchronizačního jádra existují v podobě modelu atomu sféra fyzických operací s periferiemi, sféra katalogových operací, sféra plánování procesorů a paměti, sféra inicializace virtuálních strojů, sféra zavádění kódu a dat do virtuální paměti, sféra manipulace s logickými větami datových souborů atd.
Vnější sféry se obracejí na sféry vnitřní, jazyk řízení systému může volat nezbytnou podmnožinu procedur různých sfér, event. s výjimkou procedur jádra. Na logické hierarchii sfér je založena ochrana paměti. Každá sféra má svůj číselný stupeň přístupu /nejnižší v jádře/ a součástí kódových a datových oblastí virtuální paměti je popis jejich stupně přístupu pro čtení, psaní a exekuci kódu, jenž se odvozuje z vlastností cílových modulů a stupně přístupu toho kódu, jenž oblast virtuální paměti zřizuje. Tato hierarchisace je uplatněna i v rámci uživatelského kódování a procedur jazyka řízení systému.

8. Uspořádání kódu

Mezi voláním uživatelského kódu a voláním systémové procedury není formálních rozdílů. Cílové moduly uživatelské i cílové moduly operačního systému mají totožné uspořádání a jsou v podstatě zaměnitelné. Striktně se dodržuje reentrantnost kódu. Lokální kód uživatelského VM může být užíván globálně všemi procesy smí je součástí operačního systému, musí ale mít patřičné vlastnosti /mezi nimi reentrantnost/. Ke globálně užívanému kódu přísluší lokální kopie datových oblastí.

Vývojem prošla i struktura cílového modulu: tvoří jej datové a kódové oblasti, věty popisující historii modulu, věty pro diagnostiku v případě programových chyb a věty popisující vlastnosti datových a kódových oblastí a pojmenovaných objektů v nich. Existují

prostředky pro manipulaci s cílovými moduly a jejich vlastnostmi.

9. Superstruktura

Součástí operačního systému je tzv. superstruktura produktů obsahující standardní vybavení kompilátorů a software programovacích jazyků,prostředky úpravy cílových modulů,vybavení pro údržbu datových souborů všech typů a organisací,generátory dat,prostředky ladící diagnostiky,software transakčního zpracování a databázových úloh,prostředky evidence datových struktur a programového vybavení uživatelských agend atd.

Mezi prostředky údržby souborů nutno zahrnout: systém bezpečnostních kopií datových souborů,procedury třídění a kopírování souborů,prostředky pro vytváření a údržbu knihoven cílových a zdrojových modulů včetně knihoven dat s nesekvenční organisací,prostředky pro interaktivní vstup a úpravy souborů, vybavení pro generování jednorázových sestav,diagnostické výpisy datových souborů,generátory zdrojových textů,makro- a pre- procesorovou nadstavbu jazyků.

10. Systém vstupu/výstupu

Důležitým rysem je oddělení popisu fyzické organisace a alokace datového souboru od uživatelského programu. Deskripce souboru i údaj o jeho umístění je přirozenou součástí katalogové informace o souboru. Sféra zpracování logických vět souborů se stává globální reentrantní částí operačního systému a nezvětšuje rozsah kódu uživatelského programu,který je navíc nezávislý na fyzické organisaci svých dat /v tomto smyslu si jsou ekvivalentní všechny typy přímého přístupu /. Příslušná systémová procedura,tzv. record access mechanismus,je vybrána podle deskripce souboru až v rámci otevření souboru v uživatelském programu. Takto pochopitelně funguje též celá superstruktura,pracuje-li na úrovni logických vět. Je-li otevřeno více souborů téže organisace,existuje lokálně několik kopií příslušných datových oblastí procedury / zásobníky v/v /. Existuje uživatelský interface pro řízení dat na fysické úrovni bloků mutný zvláště pro zpracování nekompatibilních magnetických pásek. Uživatel může nahradit standardní record access mechanismus svým vlastním,jehož existenci uvádí v deskripci souboru.

11. Přístupová práva

Superstruktura je organisována do několika knihoven vůči nimž jsou uživatelé seskupeni podle práva přístupu. Vztah přístupu jednoho katalogového objektu k jinému tvoří součást katalogové informace o obou objektech. Pomocí formálního rozčlenění povolených systémových funkcí do několika pseudoknihoven jsou omezována práva přístupu i v rámci sfér operačního systému,kde je navíc možno do jisté míry měnit i číselný stupeň přístupu k paměti /viz odst.7/. Takto správce systému řídí ochranu operačního systému a stejným způsobem,tj. povolováním přístupu k datovému souboru,ovládá správce dat ochranu dat. Alternativně lze tyto pravomoci delegovat jednotlivým uživatelům,což ale nemusí být žádoucí v interaktivních režimech u dat hospodářské či jiné povahy.

Aniž bychom dále doplňovali a precisovali výše popsaný výčet atributů výpočetního systému,chceme se v další části článku zaměřit na důsledky,jež popisované vlastnosti systému přinášejí do uživatelské sféry od chvíle rozhodnutí o instalaci počítače po rozběh prvních úloh uživatelského a provozního charakteru.

12. Rozhodování o instalaci výpočetního systému

Dochází-li k rozhodnutí o instalaci našeho typu počítače v organisaci kde chybí zkušenosti s obrazovkovým provozem či transakčním zpracováním,skrývají se největší úskalí v rozhodnutích o konfiguraci počítače včetně obrazovkové sítě a ve specifikaci požadovaného software. Jde-li o devízovou investici,může vést snaha o minimalisaci pořizovacích nákladů k redukcím hardware, software,dokumentace i kursů pořádaných dodavatelem. Tyto skutečnosti se bohužel brzy po rozběhu počítače projeví v malé produktivitě provozu,v podstatném snížení průchodnosti systému proti předpokladům a příliš dlouhými časy odezvy v obrazovkovém servis-u. Je nutno počítat s tím,že komplexnější operační systém,byť sám stránkovaný ve virtuální paměti,musí mít dostatečnou kvótu reálné paměti,což rozhodně bude více než kvóta garantovaná zástupci dodavatele. Rozměrové propočty potřeby vnitřní paměti na základě konkurence virtuálních strojů různých typů zpracování nutno rovněž považovat za optimistické odhady budoucí skutečnosti. Odchýlí-li se totiž obrazovkový servis od triviálního scénáře obsa-

hující operace nad nevelkými sekvenčními soubory,rostou rychle nároky na reálnou paměť a stejně rychle se prodlužuje doba odezvy. Donutí-li vnější okolnosti organisaci instalovat výpočetní systém s minimální přípustnou vnitřní pamětí,je nutno smířit se s tím,že jednotlivé typy uživatelského zpracování /dávkové úlohy, obrazovkový servis,transakční zpracování/ se navzájem prakticky téměř vylučují.

Řešením by mělo být uspořádání ověřovacích testů u dodavatele,doplněných stážemi odborníků /nikoli vyšších hospodářských pracovníků/ na existujících instalacích,i v zahraničí.

13. Projekce a programování

Od prvopočátku je nutno respektovat specifika výpočetního systému,reálné možnosti a omezení dané instalace,přičemž projektant/analytik musí být zevrubně seznámen s vlastnostmi zvoleného typu zpracování. A pozor! Znalost dávkového zpracování na počítačích druhé generace je nulovou znalostí v našem systému,či ještě spíše škodlivým návykem na překonané způsoby práce s výpočetní technikou,jež je nutno bez sentimentality opustit. Projekt by měl od prvopočátku zahrnovat návrh řídících programů úloh v jazyku řízení systému. Přirozenou cestou je užití jazyka řízení systému jako pseudokódu,v němž je návrh projektu strukturovaně rozvíjen. Přesvědčit ale projektanta,že znalost jazyka řízení systému spolu se znalostí funkcí operačního systému a poslání jednotlivých produktů superstruktury jsou nezbytnou podmínkou úspěšného projektu, přiměřeného možnostem instalace a přitom využívajícího podstatné rysy výpočetního systému,nabývá lehké;a to zvláště tehdy,je-li pojem pseudokódu v projekci dosud nezvládnutelnou novinkou. Téměř stejně obtížné bývá přesvědčit programátora,že řídící program úlohy je součástí programátorské práce a že jeho ladění je nutno věnovat nejméně tutéž pozornost co ladění uživatelských programů.

Tento fakt bývá důsledkem nepochopení skutečnosti,že operátorská aktivita a uživatelova úloha už nejsou spjaty společným záznamem na operátorově dálnopisu,ba že operátor kromě požadavků na nasazení magnetických medií a realisaci výstupů spoolingem ani nemá žádnou evidenci o průběhu úlohy. Že tedy řídící program úlohy musí automatisovat dříve operátorem plněné funkce kontroly

úspěšnosti zpracování úlohy. Teprve až se tato skutečnost stane součástí obecného povědomí,bývá akceptováno i tvrzení o nezbytnosti znát jazyk řízení systému a s ním základní systémové funkce. Programátor pak přijme konečně názor,že jen pomocí jazyka řízení systému může bez zbytku realisovat svou profesní roli a analytik připustí,že bloková struktura alokace systémových zdrojů v jazyku řízení systému lépe popisuje souvislou programovou řadu dávkového zpracování než hromádka vývojových schemat.

Podaří-li se tento okamžik posunout do doby instalace výpočetního systému,má instalace v jistém smyslu vyhráno. Profesní odborníci totiž přistupují k novému výpočetnímu systému realisticky,mají zájem ovládnout práci s novým systémem a dá se očekávat, že svým postojem příznivě ovlivní i uživatele. Pak jistě nebude nepřekonatelnou bariérou ani neadekvátnost metodiky budování ASŘ, jež je koncipována pro předchozí generace počítačů,ani nedůvěra uživatele k novinkám obrazovkového servisu.

V opačném případě lze očekávat nízkou úroveň využívání systému odůvodňovanou chybami v operačním systému,nesolidností dodavatele,neúplností dokumentace a podobně. Operační systém se stane nepřítelem,jenž ohrožuje plnění plánovaných termínů programátorských i rutinních prací. Toto povědomí naruší nezbytnou pracovní kázeň uživatelů obrazovkového servisu a přidruží-li se dočasně i hardwareové potíže,je zde reálné nebezpečí,že provoz upadne do permanentní křeče,v níž se zpracování injekcemi "léčebných" zásahů jen příškopy dostává k finálním výstupům. Takovéto stavy napětí mezi profesními složkami,provozem počítače a uživateli přispívají i k manipulačním omylům a v konečné instanci podkopávají důvěru uživatele k nové výpočetní technice.

Mnohem více než dosud záleží na včasné a kvalitní přípravě všech skupin pracovníků,jež přijdou s výpočetním systémem do styku. Za nezbytné považujeme,aby dostatečné znalosti operačního systému prokazovali i projektanti/analytici a též vedoucí profesních skupin,v jejichž pravomoci je tvorba závazných normativů provozních,programovacích a dokumentačních. Máme na mysli tyto okruhy otázek:

typy organisace dat v diskové filestore,kapacity medií a hardwareová omezení,způsoby sdílení zdrojů /media,paměť,čas procesorů/, způsoby komunikace úloh s operátorem,orientace v deníku úlohy,

povšechné znalosti o užívání systémových prostředků pro řízení filestore a katalogových operací.
Vřele doporučujeme krátkou stáž těchto pracovníků v provozu počítače,jakož i občasnou aktivní činnost u obrazovky. Praktická znalost problematiky se brzy zúročí v dvojí podobě: v kompetentních projektech analytiků a rozhodovacích aktech vedoucích a zároveň v přirozeném respektu ostatních k těmto pracovníkům.

Samozřejmě ani sebelíp vedená příprava práce nevyloučí možnost neúspěchu některého projektu. Popis řešení takovýchto situací se už vymyká námi vymezené problematice.

14. K psychologii "obrazovkového" programátora

Nároky na programování interaktivních úloh i změny související s obrazovkovým provozem programovacích prací byly publikovány. Diskutovány byly i některé aspekty kázně programátora obrazovkového typu. Dvě okolnosti však stojí za zvláštní komentář.

V prvé fázi využívání nového systému se programátor stává jediným uživatelem obrazovkového servisu. Zvyká si na svůj monopol a paradoxně se staví proti uživatelské aplikaci,kterou sám programoval,která mu ale nyní začíná ubírat část systémových zdrojů. Dožaduje se kdykoli přístupu k obrazovce a vůbec odmítá pochopit,že se může vyskytnout situace,kdy mu provoz počítače pro plné vytížení uživatelskými úlohami rutinního charakteru nemůže poskytnout ani čas v dávkovém zpracování. Tyto postoje jsou silnější tehdy,nevedou-li si vedoucí projekčních a programovacích útvarů evidenci o využívání reálného času výpočetního systému.

Druhým problémem je ladění programů v obrazovkových úlohách. Překročí-li objem dat triviální úroveň,pak současné ladění v několika úlohách způsobí podstatné zhoršení odezvy. Zákaz ladění z obrazovek likviduje podstatu interaktivní práce,navíc musí být zabezpečen softwareově,což nemusí být vždy snadné. Jediným řešením je účinný plánovací mechanismus přidělování procesorů a paměti,jenž drasticky omezí prioritu neukázněné úlohy. Tato funkce však řádně působí jen tehdy,je-li provoz počítače vyvážený vzhledem ke svým zdrojům a počtu konkurujících si aktivních úloh.

15. Extrémní aplikace

Týž problém snížení průchodnosti systému vzniká v souvislosti s exekucí extrémních úloh. Extrémně se kupodivu chovají klasické úlohy hromadného zpracování dat se sekvenčními soubory, kdy limitující není doba čtení souboru, ale nároky programu na čas procesoru, což mívá původ v nadměrných konverzích binárních, decimálních a znakových zobrazení numerických údajů v důsledku nevhodných tvarů dat jakožto pozůstatku analytického myšlení z dob děrnoštítkové techniky. Často jde o pochybné snahy o šetření na délce logických vět, což má řešit velikost souborů. Tento typ úloh obvykle minimálně stránkuje, čímž se podstatně liší od úloh s ekonomickými či vědecko-technickými aplikacemi maticového počtu. Přihlédneme-li ke způsobům pořizování vstupů v úlohách hromadného zpracování dat, stojí zřejmě před útvary projekce též zásadní změna v přístupu k návrhům formátů dat, kdy znaková representace číselných údajů může být nahrazena binárním zobrazením už při pořízení vstupů. Totéž se týká dat kmenových souborů sekvenčních i souborů databázových. Při převládajících souborech diskových zůstává sekvenční organisace ponejvíce záležitostí vstupů a klasických tiskových souborů výstupních. Rozhodujícím hlediskem pro tvar číselných údajů by měla být četnost aritmetických operací nad těmito údaji, nikoli hledisko délky údaje či věty, nehledě na to, že binární zobrazení např. v cobolu může být úspornější i z tohoto hlediska.

Aplikace maticového počtu pracují obvykle s rozsáhlými systémy lineárních algebraických rovnic a problémy spočívají v nárocích na virtuální a tím i na reálnou paměť i na čas procesoru. Limitující bývá též implementační jazyk, jehož kompilátor buď neoptimalisuje adresování, nebo jehož software neobsahuje operace vstupu/výstupu pro diskové soubory s přímým přístupem. Podle našeho názoru je zde nutno postupovat tak, že volíme implementační jazyk právě s ohledem na algoritmy adresování, vědomě se omezíme na jeho vhodnou podmnožinu a pro pracovní soubory volíme organisaci nezávisle na jazyku. Oddělení deskripce souboru od programu umožňuje totiž přímé volání systémových procedur pro zpracování logických vět souborů, čímž dává možnost vytvořit si vlastní I/O systém, eventuálně i s vlastním record access mechanismem. Nestandardně je nutno postupovat též při řešení časové otázky těchto úloh. Výchozí projekt musí jako samozřejmost předpokládat restartovatelnost

prakticky z kteréhokoli místa zpracování,což není požadavek neřešitelný. Nezajímavé není ani použití databázových struktur dat v těchto úlohách,jež podobně jako zobecněný katalogový usel řeší problém registrace postupu úlohy/algoritmu okamžitou aktualisací fysického záznamu databázové věty /katalog má charakter databáze/. Těchto prostředků lze též užít při řešení synchronisace na sobě závislých úloh.

Na tomto místě uzavíráme článek s nadějí,že naše snohdy nezdůvodňovaná tvrzení vyprovokují pozorného čtenáře k diskusi,ať už souhlasné či nesouhlasné.

Literatura:

všeobecnou problematiku operačních systémů nechť čtenář konfrontuje s knihou Hansenovou /1/ a Madnickovou /2/.
Speciální popisy systému VME/B a implementační detaily uživatelské lze studovat ve firemních manuálech ICL dostupných ve studovně útvaru správy operačních systémů VS VŽSKG v Ruské ulici v Ostravě-Vítkovicích. V přehledu uvádíme názvy důležitějších.

/1/ P.Brinch Hansen,Principy operačních systémů,SNTL 1979

/2/ Madnick,Donovan,Operační systémy,SNTL 1981

/3/ firemní literatura ICL:

SCL Vocabulary,TP6500

System Structure Manual /mnohadílná publikace věnovaná jednotlivým sférám-subsystémům oper.systému VME/B /

SCL Syntax,TP6503

Writing Job Control Programs,TP6343

System Construction and Maintenance,TP6522

Filestore Management,RP

IDMS,Part2,3,4,TP6924-26

Implementing a Transaction Processing System,TP6512

Implementing Communications System,RP

Work Management,RP

Object Module Utilities,TP6924

VME/B Project Log,soubor mikrofiší s kompilacemi procedur oper.systému a hierarchií struktur dat a jejich užití

/4/ Referenční příručka uživatelských maker a procedur,TPV001, udržovaná správou oper.systémů VS VŽSKG.