K PROBLÉMU PARALELNÍHO ZPRACOVÁNÍ

Suchomel Jiří, prom. mat.

Abstrakt: Víceprogramový operační systém, vícuživatelský programový systém, vstupně-výstupní operace, inverzní kódování, inverze programu, stavová proměnná, paralelní zpracování, synchronizace činností v programech, rozpor proložení datových struktur, stavový vektor, soubor stavových vektorů, ovladač stavových vektorů, záznam vyvolání, asynchronní periferní operace s dokončovací rutinou.

1. Úvod

V základním softwarovém vybavení výpočetních systémů existují operační systémy (OS), které dokáží zpracovávat stovky různých úloh "současně, ve stejném časovém okamžiku". Jsou to víceprogramové OS. Uživatel přistupuje k systému většinou v komunikačním režimu prostřednictvím terminálu (displeje) a po registraci specifikuje úlohu, kterou chce použít. OS vytvoří speciální kopii odpovídajícího programu, tuto kopii aktivuje a podporuje její zpracování. Pomocí displejové sítě lze necentralizovaným uživatelům poskytovat veškeré služby centrálního výpočetního systému.

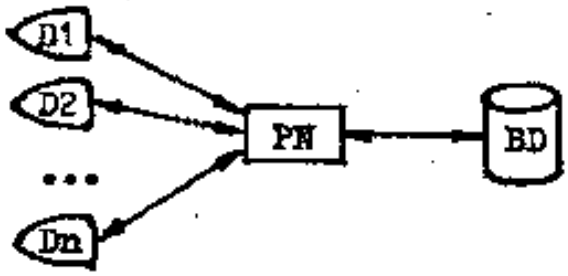
Skutečné potřeby uživatelů nejsou takto obecné, naopak, uživatelé převážně nechtějí znát nic o OS (nechtějí se mimo svou odbornost nic nového učit), chtějí pouze využívat služeb výpočetní techniky v rámci své profese, v rámci agendy, za kterou jsou odpovědní a které rozumějí. Takový uživatel dokáže odpovědět na konkrétní dotaz z obrazovky displeje : ZPRACOVANI POHYBU ZAKLADNICH PROSTREDKU ? A/N : znakem A, požaduje-li toto zpracování, ale je bezradný, má-li pod OS vyvolat program ZP33 příkazem OS : RUN ZP33 i když stokrát ví, že pro zpracování pohybů základních prostředků je určen právě program ZP33.

Uživatel tedy nevyžaduje komunikační režim na úrovni OS, ale je schopen akceptovat komunikaci na úrovni uživatelského programu a to v neposlední řadě také proto, poněvadž zde jsou mu poskytovány nápovědi pro obsluhu a je možné jeho chybné zásahy interpretovat "schovívavějším způsobem" než činí OS. Režim takové

práce uživatele je následující : Po registraci aktivuje program P, který obsahuje realizaci všech možných činností vyžadovaných uživatelem a celá následující práce uživatele je řízena programem P (obr. 1). Návrh a realizace programu P může být pouze rozsáhlá a pracná, ale určitě nebude neřešitelná.

U výpočetních systémů, kde není instalován víceprogramový OS, je uživatelský problém samozřejmě stejný, ale realizace provozu obdobného programového díla je nepoměrně obtížnější. Je nutné navrhnout a realizovat program PN (obr. 2), který musí být schopen provádět mimo funkcí programu P i určité funkce víceprogramového OS.

obr. 1



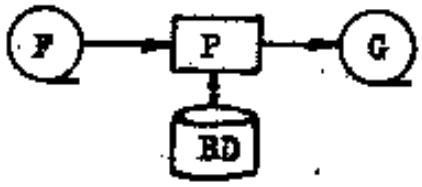
obr. 2

Problém : Je navrhnout a realizovat víceuživatelský programový systém, který nebude provozován pod víceprogramovým OS, přičemž shodný problém je vyřešen pro jediného uživatele resp. pro více uživatelů, přitom tito uživatelé přistupují k programu v nepřekrývajících se časových intervalech. Řešením problému budou v dalším odpovědi na otázky :

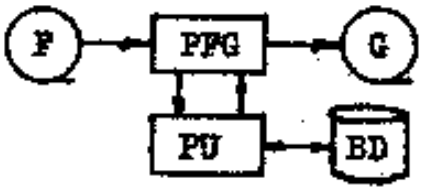
- lze použít návrh programu P pro návrh a realizaci programu PN a v jakém rozsahu ?
- jaké budou změny při přechodu od programu P k programu PN a jaká pravidla doporučit pro programovou implementaci ?
- které činnosti musí být schopen zabezpečit OS na základě našich výsledných požadavků ?
- které funkce a v jaké formě převezme program PN z víceprogramového OS ?
- v jakém formátu budou aktivovány vstupně-výstupní operace (I-O-op.) s uživatelskými displeji ?

2. Forma jednouživatelského programu

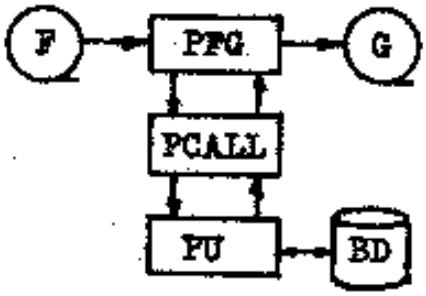
Nechť sekvenční soubor I-op. resp. O-op. je soubor F resp. G. Schema takového systému je na obr. 3. Budeme používat unifikovanou formu programové implementace. Vyčleníme všechny I-O-op. s uživatelovým displejem z programu P, tyto operace bude provádět program PFG a řešení vlastní úlohy bude provádět program PU, viz. obr. 4. Úloha bude často velmi rozsáhlá a program PU bude nutné realizovat jako množinu programových složek se segmentovou strukturou a překryvnými oblastmi. Bude vhodné rozšířit systém P o pro-



obr. 3



obr. 4



obr. 5

gram PCALL, který dokáže aktivovat libovolnou programovou složku z PU a zajistit její přímé propojení s programem PFG (obr. 5). Program PU je tedy systém podprogramů, v daném okamžiku je v operační paměti právě jediný podprogram. Program PCALL dokáže realizovat CALL volání a zpracovat RETURN v daných úrovních vyvolání. Údaje mezi podprogramy je nutné předávat přes rezidentní část programu P.

Program PFG při prvním aktivování programu PCALL požaduje vyvolání počáteční programové složky z PU a při každém dalším vyvolání vysílá informace o výsledku I-O-op. a přebírá od programu PCALL informace pro I-O-op. Program PCALL zajišťuje přítomnost dané programové složky z PU v operační paměti. Při vyvolání této programové složky z PU předává výsledek I-O-op. z programu PFG a přebírá od složky PU buď informace pro I-O-op., tyto propouští do programu PFG, nebo požadavek na aktivaci další programové složky z PU, který okamžitě realizuje na vyšší úrovni vyvolání. Logická schemata programu PFG a programu PCALL jsou v příloze 1.

Výše uvedenou formu je snadné realizovat v každém podobném uživatelském programu P. Implementace v programovacím jazyku je v inverzním kódování, program PCALL je invertován s ohledem na

propojení PFG-PCALL, programy z PU jsou invertovány s ohledem na propojení PCALL-PU. Například operaci čtení z klávesnice budeme v PU psát jako sekvenci příkazů : "kód operace := read", "pamatuj místo za exit", "exit", "místo za exit :". Operaci volání podprogramu z PU do PU (v této formě je povolena i rekurzivita) jako sekvenci : "operace := call", "volaný program := číslo programu", "pamatuj místo za exit", "exit", "místo za exit :". Při znovuvyvolání programu s těmito operacemi bude zajištěn inverzní návrat na pamatované místo. Identifikace programů z PU je vždy možná přiřazením přirozených čísel jednotlivým podprogramům. Pro rozpoznání konce práce podprogramu z PU je nutné na konci každé programové složky PU použít sekvenci : "kód operace := return", "exit".

Propojení mezi programy PFG-PCALL-PU jsou záznamy vyvolání s následující strukturou :

- komunikační oblast
  - CO kód operace, =0 open/ =1 činnost
  - CR kód výsledku, =-1 call/ =0 wait/ =1 write/ =2 read/ =3 write+read/ =4 return
- datová oblast
  - RC záznam souboru F resp. G
  - NP číslo programu, =1,2,...
  - UV úroveň vyvolání, =1,2,...,max

Programy z PU používají stavovou proměnnou QS(UV), zpracování operace "open" je přiřazení počáteční hodnoty QS(UV):=1.

Program PCALL používá stavovou proměnnou QSPC(UV). Poněvadž program PCALL je rekurzivní (rekurzivita je realizovaná uživatelsky), je nutné přiřadit počáteční hodnoty QSPC(UV):=1 pro všechny možné hodnoty UV při kompilaci a při zpracování kódu výsledku "return", t.j. při návratu na nižší úroveň, obnovit hodnotu 1 na aktuální úrovni. Počáteční hodnota UV=1.

Program PFG je hlavní program pro zpracování souborů F a G s operacemi "read F" a "write G". Záznam "eof" u souboru F dokáže rozpoznat pouze program PU, počáteční programová složka PU oznámí kód výsledku "return".

3. Problém paralelního zpracování

Máme navrhnout a realizovat program PN, který bude pod OS jako jednoprogramová úloha zpracovávat vstupně-výstupní operace z n displejů a v závislosti na těchto operacích provádět specifikované činnosti, např. nad bází dat apod. Je vypracován návrh programu P, který dokáže provádět tytéž činnosti v závislosti na výsledcích I-O-op. jediného displeje, přičemž u tohoto displeje se očekává práce více uživatelů, ale v každém časovém okamžiku může u displeje pracovat maximálně jediný uživatel.

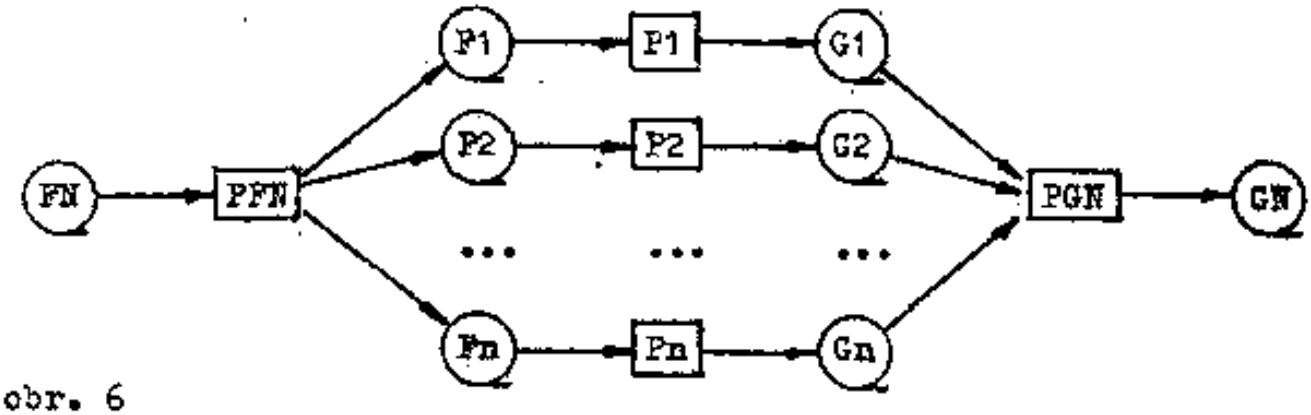
Poznámka. K obecnosti tohoto problému např. : Je navržen program pro řízení technického systému na základě zpracování výsledků měření o stavu systému (I-op.) pomocí řídících signálů (O-op.). Je požadováno řízení n takových technických systémů současně jediným programem, přitom z měření vstupují do programu pouze ty hodnoty, které jsou z daných kritických intervalů.

Těžiště řešení problému bude v synchronizaci činností v programu PN na odpovídající I-O-op. uživatelů. O synchronizaci činností v programu P bylo postaráno zadáním : 1. operace uživatele - 1. činnost programu,... Synchronizace činností v PN je nutná, poněvadž uživatelé displejové sítě nepracují ve shodném taktu a samozřejmě nelze takový způsob práce od uživatelů očekávat.

Každý i-tý uživatel (i=1,2,...,n) vytvoří při práci u displeje sekvenční soubor Fi svých vstupních zpráv, tento soubor je chronologicky setříděný. Pro zpracování souboru Fi je navržen program P, který vytvoří výstupní sekvenční soubor odpovědí Gi též chronologicky setříděný. Pracuje-li n uživatelů současně, vytvoří při své práci u displejů sekvenční soubor FN svých vstupních zpráv, soubor FN je také chronologicky setříděný, záznam je do souboru FN zapsán okamžitě po ukončení zprávy uživatelem. Soubor FN bude zpracován programem PN, program PN vytvoří výstupní sekvenční soubor GN - chronologicky setříděný - soubor odpovědí.

Zásadní poznatek je následující : V souboru FN resp. GN jsou všechny záznamy všech souborů Fi resp. Gi i-tých uživatelů a to ve stejném chronologickém pořadí, záznamy souboru Fi resp. Gi jsou pouze odděleny záznamy ostatních souborů Fj resp Gj (j≠i, j=1,2,...,n). Naše úloha návrhu programu PN je úloha řešení rozporu proložení datových struktur.

Navrhneme program PFN, který zpracuje vstupní soubor FN, viz obr. 6. Program PFN rozliší u každého záznamu z FN zdroj záznamu, tj. pořadové číslo i uživatele (displeje), a záznam zapíše do výstupního sekvenčního souboru Fi. Všechny soubory Fi pak zpracujeme programem Pi, který vytvoří výstupní sekvenční soubory Gi. Souborů Fi, Gi bude tolik, kolik displejů ze sítě použili uživatelé ke komunikaci. Programy Pi budou totožné, bude to program P. Navrhneme program PGN, který zpracuje všechny soubory Gi a jejich záznamy předá na obrazovky příslušných displejů - soubor GN.



obr. 6

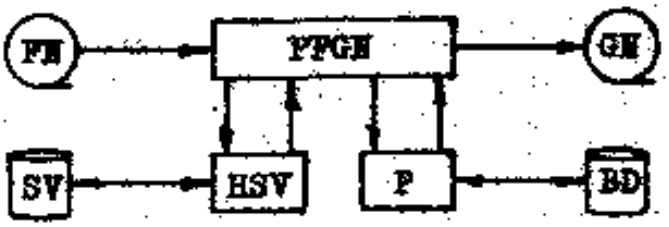
Problém je vyřešen, ale za jakou cenu. Musí fyzicky existovat soubory Fi a Gi a jejich zpracování je časově odtrženo od zpracování souboru FN a od vytvoření souboru GN. Takové řešení není přijatelné u naší úlohy, žádný uživatel nebude nikdy schopen zadat dopředu všechny své požadavky, hlavně proto, poněvadž většinu svých činností na klávesnici displeje volí podle výsledků na obrazovce displeje svých předcházejících požadavků. U řízení technických systémů bude zase požadována okamžitá reakce na nenormální stavy systémů a náš programový systém se nemůže chovat jako generál po bitvě.

V systému PN musí být posloupnost zpracování následující : Při obdržení zprávy od i-tého uživatele musíme okamžitě na tuto zprávu reagovat zpracováním v programu P a výslednou odpověď okamžitě odeslat na obrazovku i-tého uživatele. Uživatel může psát svou zprávu pouze na základě operace read programu PN. Program PN může zahájit zpracování zprávy uživatele až po ukončení operace read, zpracování končí vystavením operace write - program PN odesílá odpověď a další operaci read může program PN aktivovat až po ukončení operace write. Pro řízení práce programu PN jsou proto

důležité okamžiky ukončení I-O-op. s displejem. Poměr času provádění periferních operací a času zpracování výsledků těchto operací určuje akceschopnost programu PN, poměr by měl být větší než n.

Programy PFN a PGN spojíme do jediného programu PFGN, který dokáže rozpoznat a zpracovat příznak ukončenosti periferních operací u všech displejů v síti. Danou ukončenou periferní operaci předá program PFGN ke zpracování příslušné verzi programu P. Zpracování je okamžité a výsledkem je odpověď pro příslušného uživatele, odpověď je vyslána na obrazovku a program PFGN je informován, že na příslušném displeji se provádí periferní operace. Další ukončenou periferní operaci nechá program PFGN zpracovat ...

Při programové implementaci programu PN je nutné invertovat program PFG s ohledem na propojení mezi programy PFGN-PFG, jsou to příznaky "end of record" ze souborů Fi a Gi. Příslušnou verzi programu P budeme realizovat pomocí stavového vektoru programu P.



obr. 7

Stavový vektor musí obsahovat všechny proměnné specifické pro příslušný displej. Stavové vektory budeme uchovávat v souboru stavových vektorů SV na vnější paměti s přímým přístupem. Program PFGN bude manipulovat se stavovými vektory pomocí ovladače stavových vektorů HSV (obr. 7).

Abychom mohli využít čas nutný k provedení periferních operací pro zpracování výsledků periferních operací ostatních uživatelů, musíme používat pro displeje asynchronní periferní operace. Synchronní periferní operace jsou nepoužitelné, není myslitelné aktivovat např. read a čekat se zpracováním PN na ukončení read, uživatel může mít v tomto čase nutnější práci, než odesílání zprávy. Všechny operace s displeji v PN budeme psát ve formátu : "startuji periferní operaci (vstupní či výstupní) a nečekám na její ukončení, ale jsem kdykoli připraven přijmout zprávu o jejím ukončení". Řízení práce programu PN takto přebírá vstupní-výstupní systém OS, který vytváří sekvenční soubor signálů o ukončených periferních operacích. Výhodné je používat asynchronní periferní operace s dokončovací rutinou. Dokončovací rutina je uživatelský podprogram, tento podprogram je aktivován operačním systémem při ukončení periferní operace.

4. Odpovědi

Návrh programu P pro realizaci programu PN lze použít v plné šíři. Dokonce můžeme požadovat pro realizaci programu PN jako postačující podmínku právě návrh jednouživatelského programu P. Pak program P doplníme na program PN programem PFGN - jeho logické schema je v příloze 1.

Bude-li mít programová implementace P unifikovanou verzi, změní se při přechodu od P k PN pouze zápis programu PFG. Program PFG bude invertován s ohledem na propojení PFGN-PFG. Toto propojení budeme realizovat rozšířením stávajícího záznamu propojení v datové oblasti o položky : ND - číslo aktuálního displeje, SD(ND) - okamžitý stav displejů =0 nepracuje/ =1 pracuje.

U programu P musíme určit stavový vektor tak, aby platilo : P je programová složka proměnného stavu a její činnost musí být jednoznačně určena
- textem programu P
- záznamem vyvolání
- stavovým vektorem.

Text programu je konstantní, záznam vyvolání přichází z vně, pouze stavový vektor uchovává stav zpracování po předešlých vyvoláních pro jednotlivé displeje. Stavový vektor musí obsahovat :
- QSPFG - stavová proměnná programu PFG
- QSPC (UV) - stavová proměnná programu PCALL
- QS (UV) - stavová proměnná programů z PU
- STOPA(UV) - trasa vyvolání programů z PU
- UV - aktuální úroveň vyvolání
- všechny proměnné z PU specifické pro daný displej.

Stavových vektorů bude stejný počet jako displejů v síti a poněvadž lze přiřadit N displejům logická čísla ND=1,2,...,N, lze chápat soubor stavových vektorů jako pole s indexem ND. V zápisu programu P opatříme indexem ND všechny proměnné ze stavového vektoru (např. "STOPA(UV(ND),ND)"). Stavové vektory budou uloženy v rezidentní části programu PN. Obecně je nutné ukládat soubor stavových vektorů na vnější paměti s přímým přístupem a realizovat program HSV pro práci se stavovými vektory.

Operační systém musí být schopen zabezpečit asynchronní periferní operace s dokončením nezávislým na zpracování programu.

Program PN (konkrétně PFGN) přebírá z víceprogramového OS činnost aktivace uživatelovy verze programu P na základě ukončení jeho periferních operací.

Periferní operace s displeji bude aktivovat jedině programová složka PFG. Takové řešení je výhodné, jinak bychom museli psát nestandardní read-write v daném programovacím jazyku (vyjma assembleru). Formát periferní operace pro ND-tý displej v PFG je následující :

```
                    SD(ND) := 1 ;
                    start periferní operace, nečekám na ukonče-
                        ní, ale očekávám SD(ND) = 0 až operace
                        bude ukončena ;
                    pamatuj místo za exit ;
                    exit ;
místo za exit :
```

5. Praxe

Na výše uvedeném principu je v současné době realizován v ORGAPROJEKTu Praha automatizovaný systém řízení a evidence výroby na výpočetním systému SMĚ-10 s displejovou sítí (4 displeje). Použitý jazyk je FORTRAN, velmi vyjímečně MACRO ASSEMBLER, operační systém FOBOS. Uživatelský systém je průběžně doplňován o další činnosti, v současné době je v systému cca 230 programových složek, kterýkoli uživatel si může nezávisle na ostatních zvolit jakoukoli činnost podporovanou těmito programy. Praktické zkušenosti jsou velmi dobré, systém je určitě akceschopnější, než táž úloha provozovaná na stejném výpočetním systému pod vyšším, víceprogramovým operačním systémem DOS RV.

Zvolené řešení je vhodné jak pro údržbu stávajících programů, tak pro snadné doplňování programů nových. V příloze jsou uvedena logická schemata řídících programů. Pro nedostatek místa nebylo možné uvést příklad v programovacím jazyku.

Literatura : M. Jackson : Principles of Program Design
Londýn 1975
ÚVT TESLA : Knihovna - FOBOS - SMEP
Praha 1978 - 1981

Příloha 1 : Logická schemata

```
PFGN    seq
           proveď počáteční práce s BD;
           SD(i):=-1, i=1,...,N;
PFGNB      itr until konec práce
              ND:=1;
DISPLEJ       itr until ND N
OPEN             sel SD(ND)=-1
                    "open" PFG;
                    "ulož" SV(ND);
OPEN             end
OPERACE          sel SD(ND)=0
                    "zjisti" SV(ND);
                    "činnost" PFG;
                    "změn" SV(ND);
OPERACE          end
CLOSE            sel CR=4
                    "zruš" SV(ND);
CLOSE            end
                 ND:=ND+1;
DISPLEJ       end
PFGNB      end
           proveď ukončující práce s BD;
PFGN    end

PFG     seq
           SD(ND):=0;
           inicializace displeje;
           NP:=1;
           "činnost" PCALL;
PFGB       itr until CR=4
OPERACE       sel CR=3
                 "write";
                 "read";
OPERACE       or  CR=1
                 "write";
OPERACE       or  CR=2
                 "read";
OPERACE       or  CR=0
                 "wait";
OPERACE       end
              "činnost" PCALL;
PFGB       end
           SD(ND):=-1;
           "return";
PFG     end

PCALL   seq                          RESULT  sel CR=-1
           STOPA(UV):=NP;                       UV:=UV+1;
           "open" PU;                           do PCALL;
           "činnost" PU;             RESULT  or
PROGRAM    itr until CR=4                       "operace" pro PFG;
              do RESULT;             RESULT  end
              "činnost" PU;
PROGRAM    end
           UV:=UV-1;
PCALL   end
```