S Y S T É M Ř Í Z E N Í Ú L O H J C S

RNDr. Jiří Boleslav - ČSTSP, ZVT VS Brno

1. Úvod

Systém řízení úloh JCS (Job Control System) představuje softwarový produkt, usnadňující uživatelům operačního systému OS tvorbu flexibilních úloh. Může spolupracovat s libovolnou variantou systému OS, tj. jak s PCP, MFT a MVT, tak i s VS 1, VS 2 a VM, případně SVS a SVM. JCS je tvořen soustavou modulů a množinou pravidel pro jejich vhodnou aplikaci. Na rozdíl od některých jiných systémů, určených pro podobné účely (viz např. [1]), pracuje JCS interpretačním způsobem.

Hlavním cílem, který byl při tvorbě tohoto systému sledován, bylo umožnit uživatelům vytváření úloh, které by bylo možno ovládat pružnějším způsobem, než operační systém OS standardně dovoluje. Tento požadavek vyplynul především z charakteru úloh, zpracovávaných v ČSTSP, v nichž je často zapotřebí zpracovávat pouze určité kontinuální úseky úloh, případně jejich jednotlivé kroky. Jazyk řízení úloh JCL operačního systému OS není bohužel přes svou značnou komplikovanost i rozsah pro podobné účely vybaven zrovna nejlépe. Neobsahuje totiž žádný příkaz skoku a jediný prostředek, umožňující větvení úloh, představuje parametr COND příkazu JOB, resp. EXEC. Další skutečností, která silně ovlivnila naše rozhodnutí, byl fakt, že v inkriminované době (tj. asi před 1 1/2 rokem) nefungoval u nás v instalované verzi operačního systému OS odložený step-restart. Bylo proto rozhodnuto vytvořit systém, který by v sobě zahrnoval jak funkci odloženého step-restertu, tak i dříve popsané možnosti větvení výpočtů.

2. Požadavky na systém

Při realizaci systému byl kladen důraz na následující požadavky:

a) Spolehlivost. Požadavek spolehlivosti a správnosti softwarových produktů vystupuje v poslední době stále naléhavěji do popředí a podle našeho názoru je tento trend zcela na místě. Sebehezčí program je pro praktické účely naprosto bezcenný, nevykazuje-li

kromě své krásy i odpovídající procento nutné spolehlivosti. Kromě toho jsou nespolehlivé programy podobné nezvedným dětem; člověk se musí o něž rovněž neustále starat, jsou zdrojem stálých potíží a konfliktů a pokud se jejich autorům nepodaří je zavčas umravnit, přerostou jim záhy přes hlavu.

b) Malý rozsah. Tento požadavek byl motivován řadou důvodů, z nichž nejzávažnějším byla skutečnost, že bylo třeba systém realizovat v době co možná nejkratší. Další důvod byl ten, že práce na JCS probíhala paralelně s prací na jiných úkolech a nebylo tudíž možné jí věnovat příliš velkou programátorskou kapacitu. Dalším faktem, který mluvil pro malý rozsah systému, byl už dříve uvedený požadavek spolehlivosti. Je totiž známo, že počet chyb v programových systémech roste s jejich rozsahem mnohem rychleji než lineárně.

c) Snadná aplikace. Požadavek snadné aplikace byl dán především tím, pro koho byl systém určen. Hlavními uživateli systému jsou aplikační programátoři a pracovníci provozu VS, u nichž nelze předpokládat ochotu absorbovat mimo několika základních znalostí o operačním systému navíc nějakou jeho příliš složitou nadstavbu nebo modifikaci.

d) Portabilita. Podobně jako spolehlivost hraje i otázka přenositelnosti softwarových produktů v současné době stále významnější roli. Málokterá instituce si totiž může při přechodu na jiný typ počítače dovolit přeprogramovávat i celý svůj uživatelský software. ČSTSP přechází v současné době z počítačů řady JSEP-1 na JSEP-2. V našem případě jsme se proto omezili pouze na portabilitu v rámci operačního systému OS.

Všechny výše uvedené požadavky byly brány v potaz při řešení základní otázky, kterou představoval způsob realizace daného systému. V podstatě v tomto případě, tj. v případě, že standardní možnosti operačního systému nejsou pro řešení určitého problému zcela vyhovující, existují vždy dvě možnosti cesty:

1. Úprava stávajícího operačního systému, resp. jeho příslušné komponenty (v našem případě kompilátoru JCL).
2. Vytvoření další vrstvy operačního systému, v našem případě formou nadstavby nad JCL.

Oba tyto postupy mají své výhody i nevýhody. Domníváme se však, že ve většině případů je vhodnější druhý postup, zatímco

první je rozumné použít hlavně v případech, kdy druhý z nějakých důvodů aplikovat nelze, tedy například v případě chyb v systému. První metoda totiž předpokládá jednak hlubší znalost systému než je v podmínkách většiny výpočetních středisek možné, jednak bývá značně problematická z hlediska portability i na úrovni verzí operačního systému. Navíc její použití končí obvykle tím, že se její uživatel stane nekompatibilním se všemi ostatními. S tímto postupem učinilo naše výpočetní středisko negativní zkušenosti již v době provozování operačního systému DOS a chtěli jsme se ho proto nyní vyvarovat.

Z výše uvedených důvodů jsme se rozhodli aplikovat druhou alternativu, tj. použít nadstavby nad JCL, která by využívala a dále rozvíjela standardní možnosti řízení úloh na základě vratného kódu a parametru COND příkazu JOB resp. EXEC.

3. Komponenty JCS

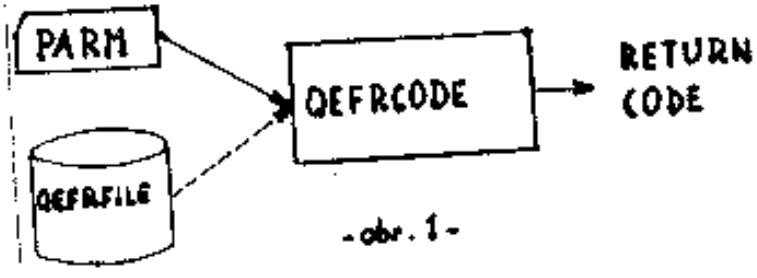
Systém JCS sestává z následujících komponent:
- modulu QEFRCODE
- modulu QEFURC
- modulu QEFABND
- pravidel pro tvorbu úloh.

Bližší popis jednotlivých komponent je obsažen v dalších odstavcích.

3.1. Modul QEFRCODE

Účel a funkce

Modul QEFRCODE je určen pro řízení úloh na úrovni JCL, tj. na základě informací, známých v době zadávání úlohy. Jeho základní funkce spočívá ve tvorbě vratného kódu na základě vstupní informace, zadané v parametru PARM příkazu EXEC, případně ze použití souboru QEFRFILE. Způsob práce modulu lze vyjádřit následujícím diagramem:



-obr.1-

Podle tvaru a hodnoty parametru PARM existují tři možné režimy práce modulu:

1. Je-li obsahem parametru PARM číselná hodnota, vrátí modul tuto hodnotu (je-li v intervalu 0 - 4095), případně ohlásí chybu (je-li mimo tento interval).
2. Není-li hodnota PARMu numerická a nezačíná znakem '*', prohlíží QEFRCODE tabulku QEFRTAB, která je jeho součástí; nenalezne-li řetězec znaků, zadaný v PARMu, na levé straně této tabulky, vrátí QEFRCODE příslušnou číselnou hodnotu, nacházející se na pravé straně této tabulky. Nenalezne-li se příslušný řetězec v tabulce, ohlásí modul chybu. Standardně je QEFRTAB tvořena jmény jednotlivých procedur ladícího systému SLS, který je v našem VS provozován a jim odpovídajícími vratnými kódy. Tuto tabulku však může uživatel nahradit libovolnou vlastní; v tom případě je třeba program QEFRCODE znovu přeložit.
3. Je-li hodnota PARMu nenumerická, začínající znakem '*', otevře QEFRCODE soubor QEFRFILE a hledá příslušnou informaci v něm obdobným způsobem, jako v bodě 2 v tabulce QEFRTAB. Tento režim práce modulu je vhodný pro případy, kdy je daná tabulka příliš dynamická na to, aby se vyplatilo vždy před spuštěním modulu provádět jeho rekomplikaci. Je třeba si ovšem uvědomit, že tento režim práce je podstatně časově náročnější než oba předchozí.

Vyvolání modulu

Modul QEFRCODE lze vyvolat následujícími příkazy JCL:

```
//   EXEC  PGM=QEFRCODE,  PARM = 'RC'
//   SYSOUT  DD  SYSOUT = A
//   QEFRFILE  DD  ...
kde  0 ≤ RC ≤ 4095        pro režim 1/
     RC = 'XXXX'          pro režim 2/
     RC = '* YYYY'        pro režim 3/
```

3.2. Modul QEFURG

Účel a funkce

Modul QEFURG slouží k alternativnímu výběru určitých kroků úlohy. Na základě informace, zadané v parametru PARM, případně informace od obsluhy, generuje modul vratný kód 0, resp. 1. Schéma

jeho zpracování vyjadřuje následující diagram:

- obr. 2 -

Podle hodnoty, zadané v parametru PARM, pracuje modul dvěma různými způsoby:

1. Je-li v PARMu uvedena některá z následujících hodnot:
   - ANO, A, YES, Y, generuje modul vratný kód 1
   - NE, N, NO , generuje se vratný kód 0.
2. Je-li v PARMu uveden nějaký jiný neprázdný řetězec znaků, vypíše ho QEFURG na konsolu a čeká na odpověď operátora; s ní potom nakládá analogicky, jako v bodě 1/ s hodnotou PARMU. V případě prázdného PARMu generuje QEFURG vratný kód 0.

Z předchozího popisu je zřejmé, že prvá varianta použití modulu je určena pro řízení úloh na úrovni JCL, tj. na základě informací, známých v době jejich zadání. Naopak druhý způsob je určen pro řízení úloh na operátorské úrovni, tj. pro řízení na základě informací, které jsou k dispozici až v době jejich zpracování. Je zřejmě vhodné vždy preferovat prvou metodu a druhou používat pouze v případě nezbytné nutnosti.

Příklad

Může se stát, že určitá informace (např. nutnost zpracování jistého souboru) může, ale nemusí, být známa již před provedením dané úlohy. Proceduru pro zpracování takové úlohy lze pak definovat následujícím způsobem:

```
//PRIKLAD     PROC   ....,
//            BEH1 = 'ABC103D   BUDE VSTUP    PO 1  ? ',
                :
                .
//ST1X        EXEC     PGM = QEFURG, PARM = '&BEH1 '
//SYSPRINT     DD      SYSOUT = A, SPACE = (121,(5))
//ST1         EXEC     PGM = XY2, COND = (0,NE,ST1X)
```

Je-li potom informace o existenci daného souboru známa již před provedením úlohy, lze kódovat např.:

// EXEC PREKLAD, BEHI = 'A', ...

a krok STI bude proveden, aniž by měl operátor možnost o jeho spuštění rozhodnout. Nebude-li naopak parametr BEHI při volání procedury použit, obdrží operátor dotaz ABC 103 D a musí na něj odpovědět.

3.β. Modul QEPABND

Modul QEPABND slouží k abnormálnímu ukončení úlohy. Při zpracování úloh dochází totiž často k situacím, kdy je třeba jejich chod abnormálně ukončit. Poněvadž JCL ani vyšší programovací jazyky nejsou pro tyto účely vybaveny žádnými vhodnými prostředky, byl v rámci VS vytvořen modul QEPABND, eliminující tento nedostatek.

Použití v JCL

Na úrovni JCL lze modul vyvolat příkazem:

```
//      EXEC    PGM = QEPABND,  PARM = 'U, TEXT'
//SYSPRINT DD  SYSOUT = A
```

kde

U je číslo uživatelského ABENDu v intervalu od 0 do 4095 a TEXT je řetězec znaků v délce do 80 B, který se vypisuje na operátorské konsole a v souboru SYSPRINT; standardní hodnota U = 3.

Použití modulu QEPABND na úrovni JCL je vhodné především ve spojitosti s programy, které i v případě neúspěšného ukončení své činnosti nekončí abnormálně, tedy např. v případě služebních programů.

Použití v COBOLu

V jazyku COBOL lze vyvolat příkazem:

```
CALL 'QEPABND 'USING POLE
```

kde POLE má následující formát:

```
01  POLE
  05  DELKA  PIC  9(2)  COMP  VALUE nn .
  05  PARM   PIC  X(nn) VALUE 'U, TEXT'.
```

Přitom nn vyjadřuje délku řetězce PARM v bytech a U a TEXT mají tentýž formát i význam jako v předchozím odstavci.

Použití v PL/1

V jazyku PL/1 lze modul vyvolat pomocí příkazů:

```
% INCLUDE QPLABND;
CALL  QPLABND ('U, TEXT');
```

kde U a TEXT mají opět stejný význam jako v předchozích odstavcích. Navíc je třeba mít v knihovně maker PL/1 uložen text QPLABND a kompilovat s volbou překladače MACRO.

3.4. Pravidla pro tvorbu úloh

JCS usnadňuje uživateli tvorbu flexibilnějších úloh než je standardně v OSu možné. Při tvorbě těchto úloh je třeba dodržovat následující pravidla:

1. Všechny úlohy, používající systém JCS, je nutno psát ve tvaru procedur jazyka JCL.
2. Ke každému kroku úlohy je třeba přiřadit určitou hodnotu návratného kódu, a to rostoucím způsobem od prvého kroku k poslednímu. Přírůstek těchto hodnot je vhodné volit konstantní.

   1. Tím je usnadněna možnost dodatečného vkládání nových kroků. V praxi se nám osvědčila volba přírůstku = 10. Dále je rovněž vhodné, ne však nutné, jednotlivé kroky odpovídajícím způsobem pojmenovat, tj. např. S10, S20, S30 atd. Tekové pojmenování je vhodné, pracujeme-li bez použití tabulky QEFRTAB resp. souboru QEFRFILE.
3. V každém kroku rozšířit parametr COND o podmínky
   ..... (a, LT, FROM), (a, GT, TO), ...
   kde a je hodnota vratného kódu, příslušná k danému kroku podle bodu 2/, tedy např.:

```
//S40     EXEC      PGM = XYZ
//                  COND = ((40, LT, FROM), (40, GT, TO))
```

4. Na počátku procedury provést dvoje vyvolání modulu QEFRCODE ve tvaru:

```
//FROM      EXEC      PGM = QEFRCODE, PARM = '&FROM'
//SYSOUT      DD      SYSOUT = A
//TO        EXEC      PGM = QEFRCODE, PARM = '&TO'
//SYSOUT      DD      SYSOUT = A
```

5. V záhlaví procedury (příkaz PROC) nastavit standartní hodnoty parametrů FROM a TO. Tato podmínka sice není bezprostředně

nutná, obvykle je však vhodné tak učinit. Pravidla, uvedená v bodech 1/ až 5/, umožňují uživateli při vyvolání výběr určité sekvence kroků procedury pomocí parametrů FROM a TO. Např. kódováním:

//FROM = 30, TO = 50

docílíme zpracování pouze 3., 4. a 5. kroku určité procedury. Chceme-li navíc použít možnosti alternativního výběru jistých kroků, můžeme k tomuto účelu použít modul QEFURC. V tomto případě je pak třeba:

6. Před provedením příslušného programu umístit volání modulu QEFURC ve tvaru:

   //ALT EXEC PGM = QEFURC, PARM = 'ALT1'

7. V příkaze EXEC příslušného programu rozšířit parametr COND o podmínku (0, EQ, ALT).
8. V záhlaví procedury nastavit standardní hodnotu parametru ALT1, např. ALT1 = ANO.

Všechna výše uvedená pravidla umožňují (spolu se standardní tabulkou QEFRTAB) například vytvoření procedury COBOL, kterou lze vyvolat příkazem:

// EXEC COBOL, FROM = UPD, TO = LNK, RT = Y, M = PROG

Tento příkaz znamená, že je třeba provést editaci textu jménem PROG, jeho kompilaci a linkování a navíc má být před kompilací spuštěn překladač rozhodovacích tabulek.

4. Přínosy řešení

JCS poskytuje uživateli ve srovnání s klasickou metodou zadávání úloh ve formě děrných štítků, které se ještě bohužel v řadě výpočetních středisek používá, celou řadu předností. Uveďme z nich alespoň:

1. úplnou, případně částečnou eliminaci zadávaných štítků JCL, a tím i usnadnění manipulace s nimi
2. zvýšení spolehlivosti vlivem eliminace lidského faktoru
3. úsporu strojového času
4. úsporu času pracovníků v oblasti přípravy zpracování

Tyto úspory se projeví především v oblasti zpracování rutinných úloh. JCS lze však s výhodou použít i při ladění programů, kde namísto obvykle používaného většího množství speciálních procedur pro editaci, překlad, linkování a spuštění programů (pří-

padně některé další funkce), lze vytvořit pro každý jazyk pouze jednu univerzální proceduru. Toto řešení skýtá uživateli následující výhody:

5. úsporu diskové kapacity v knihovně procedur a tím i strojového času
6. úsporu programátorských kapacit při tvorbě nových procedur
7. úsporu strojového času a materiálu, potřebných na jejich ladění.

Obě posledně uvedené výhody se v našem středisku zvlášť výrazně projevily při přechodu na ladicí systém SLS. V neposlední řadě vidíme značnou výhodu JCS i v tom, že

8. k vyřešení z metodického hlediska blízkých problémů vystačí uživatel se znalostí a užíváním jediné univerzální procedury namísto celé třídy procedur speciálních.

Naopak jistou nevýhodou JCS ve srovnání s jinými systémy může někdy být jeho interpretační způsob práce.

5. Závěr

Závěrem lze říci, že na základě zkušeností, které jsme se systémem JCS za téměř dvouletou dobu jeho používání v ČSTSP učinili, představuje tento systém velmi vhodnou pomůcku jak pro pracovníky provozního, tak i programovacího úseku. S jeho zavedením ani provozem nebyly téměř žádné problémy a v současné době představuje již v podstatě neodmyslitelnou součást operačního systému, v ČSTSP provozovaného.

Literatura

1. J. Kaniok: Generování příkazů JCL v operačním systému OS, sborník Programování 81, str. 280-287