T E X T O V Ý E D I T O R J A K O N Á S T R O J A K T U A L I Z A C E D A T O V Ý C H S O U B O R Ů

Jaroslav Pazdziora

Nasazení editoru na datový soubor není vždy přímo možné. Ukazuje se však, že v některých případech existuje uspokojivé řešení. Protože editor dovoluje změnit v souboru cokoli, je jeho izolované použití nebezpečné. Východiskem z této situace je pořízení protokolu o provedených změnách.

1. První kroky v interaktivním zpracování

Poprvé se programátor setká s novými možnostmi, které sebou přinášejí prostředky interaktivního zpracování, ve fázi přípravy nových programů. Každý výpočetní systém inzerovaný jako systém podporující interaktivní práci zahrnuje textový editor - programový prostředek pro zápis a změnu zdrojových textů programů. Pokud programátor začíná s interaktivní prací po létech dávkového zpracování, upadá nejprve do stavu euforie nad snadností, s jakou lze provádět úpravy v programech - změna jediného znaku již nezname nená nutnost přepsání celého řádku programu. Přestože po počátečním nadšení následuje vystřízlivění v důsledku nových problémů, jako je např. přijatelná doba odezvy, pružnost textového editoru září jako stálice.

Budoucnost programátora v oblasti zpracování hromadných dat je v tvorbě dialogových programů. Přítomnost však spočívá v údržbě programů pracujících v dávkovém režimu a dávkový režim bude také často prostředím, ve kterém budou pracovat programy, které programátor nově (už interaktivně) tvoří. Naskýtá se otázka, zda by nešlo využít nesporných výhod interaktivního zpracování při zabezpečení alespoň některých kroků úloh koncipovaných dávkově.

2. Výchozí podmínky

Mnohé úlohy zajišťované pracovištěm autora nezačínají fází pořízení dat, ale kontrolou sekvenčních souborů předaných do výpočetního střediska na magnetických páskách. Fáze pořízení proběhla jinde - často na jiném konci republiky - a různým způsobem: pořízením na záznamnících klávesnice - disk, optickým snímáním textů

psacím strojem písmem OCR-B či jako výsledek automatizovaného zpracování v jiném výpočetním středisku. Kontrolní programy odhalí formální chyby, další požadavky na změny doplní uživatel jako výsledek intelektuálního zpracování. Včasné provedení požadovaných změn v dávkovém režimu je problém principiálně neřešitelný.

Dále popisujeme praktické zkušenosti s textovým editorem realizovaným příkazem EDIT v operačním systému IBM OS MVT/TSO 21.8. na počítači EC 1033 s terminály EC 7906. Domníváme se však, že mají obecnější platnost. Ověřili jsme si to na minipočítači Wang 2200, kde spojení textového editoru s interaktivním překladačem jazyka Basic vedlo k nutnosti transformace datového souboru do tvaru programů tvořených jediným typem příkazů - příkazem poznámky.

3. Editovaný soubor

Soubor, na který aplikujeme textový editor, budeme nazývat editovaným souborem. Velkou předností TSO je, že editovaným souborem může být obecný sekvenční soubor, pokud splňuje omezující podmínku v délce věty. Textový editor má řádkový charakter. Na obrazovku terminálu zobrazuje větu editovaného souboru jako posloupnost znaků, které zaplní jeden či více řádků obrazovky. Ztotožnění aktualizovaného datového souboru s editovaným souborem, tj. nasazení textového editoru přímo na aktualizovaný soubor, vede k úspěchu pouze ve speciálních případech. Obecně není zobrazená věta čitelná buď proto, že splývají hodnoty sousedních položek, nebo protože položky jsou uložené ve větě kódovaně. Tak je tomu s poslední pozicí položky v zónovém tvaru se znaménkem nebo s položkou v dekadickém zhuštěném tvaru.

Problematika zobrazení datové věty na obrazovce je v telekomunikačních monitorech typu IDMS/DC řešena procesem zvaným mapování obrazovky. Při něm jsou kódované položky převedeny do čitelného tvaru, položky na obrazovce jsou oddělany mezerami a pomocnými vysvětlujícími texty, pořadí položek je změněno tak, aby výsledek byl co nejpřehlednější. K této transformaci dochází dynamicky v okamžiku zpřístupnění datové věty.

Mapování nahradíme jednorázovým přepisem aktualizovaného souboru do pracovního souboru, který bude sloužit jako editovaný soubor. Program, který tento přepis provádí (dále jej budeme

označovat jako program A), je přepis sekvenčního souboru. Je ovšem závislý na aplikaci, tj. na editovaném souboru. Závislost je však omezena na deklaraci vstupní a výstupní struktury. V jazyce PL/1 vstupní struktura U a výstupní struktury V vypadají např. takto:

```
DCL 1 U, 2 VH DEC FIXED(5), 2 VO DEC FIXED (3), 2 VZ DEC FIXED (7),
2 SOB DEC FIXED(3), 2 SP DEC FIXED(7,1), 2 NNN DEC FIXED(7), 2 NNN1
DEC FIXED(1); DCL 1 V, 2 TXT1 CHAR(3) INIT 'CP:', 2 VH PIC '9999',
2 TXT2 CHAR(6) INIT ' VOC: ', 2 VO PIC '999', 2 TXT3 CHAR (9)
INIT ' VYROBA: ' , 2 VZ , PIC 'ZZZZZ9S', 2 TXT4 CHAR(6) INIT 'A
MAT: ', 2 SOB PIC'999', 2 TXT5 CHAR (1Ø) INIT ' SPOTREBA: ',
2 SP PIC 'ZZZZ9.V9S',
2 TXT6 CHAR(19) INIT 'B          KAT: ', 2 NNN1 PIC'9',
2 TXT7 CHAR(1)  INIT 'C';
```

Přesun ze vstupní do výstupní struktury a zároveň potřebnou transformaci realizuje příkaz V=U, BY NAME; V uvedeném příkladu bude datová věta na obrazovce vypadat takto:

```
CP:12Ø1  VOC:123  VYROBA:  1234+A MAT:456 SPOTREBA: 9876.5+B ...
```

Alfanumerické položky zobrazujeme na obrazovce zarovnané doleva, numerické zarovnané doprava s potlačením levostranných a zachováním pravostranných nul, se znaménkem za poslední číslicí. Za numerickými hodnotami uvádíme písmena abecedy jako pomůcku pro identifikaci měněné hodnoty. Představme si např., že třetí nulový údaj na řádku chceme zaměnit hodnotou 55. Provedeme to příkazem

```
CHANGE * X Ø+CX55+CX
```

Vkládané pomocné texty volíme co nejstručnější, abychom nepřesáhli omezení kladené na délku věty editovaného souboru. Jinak bychom museli programem A přepsat jednu větu aktualizovaného souboru do více vět editovaného souboru. Při vkládání nových vět není nutné pomocné texty vypisovat, je možné je nahradit mezerami.

4. Bezpečnost prováděných operací

Protože textový editor má řádkový charakter, nerespektuje hranice mezi jednotlivými položkami. Je proto nutné pečlivě dodržovat pozice položek na řádku obrazovky. Příkaz CHANGE bychom měli používat se stejnou délkou staré a nové hodnoty. Nasazení textového editoru do rutinního provozu však vyžaduje nástroj, který zmenší pravděpodobnost zavlečení chyb do aktualizovaného souboru. Tímto

...rojem je program (dále označovaný jako program B) vytvářející protokol o provedených změnách. Označme editovaný soubor S. Po ukončení práce řízené příkazem EDIT uložíme novou verzi souboru do nového souboru N. Program B zpracovává souběžně soubory S a N. Provádí párování vět obou souborů a rozlišuje čtyři možnosti: spárované věty se neliší ani v datových položkách, spárované věty se v datových položkách liší, v souboru S existuje věta neexistující v N (tato věta byla při editaci zrušena) a konečně v souboru N existuje věta neexistující v S (tato věta byla při editaci přidána).

Párování vět probíhá podle klíče sestaveného z jedné či více položek editovaného souboru. Pokud editovaný soubor obsahuje jednoznačný klíč a soubor je podle tohoto klíče setříděn, je párování triviálním problémem. V našem případě však pracujeme se soubory, v nichž existují klíče, soubory však nejsou podle nich tříděné a připouští se duplicita výskytů klíčů. Naštěstí je možné každou aplikaci na našem pracovišti zařadit do jedné ze dvou skupin: buď je možno předem odhadnout maximální počet k po sobě jdoucích vět editovaného souboru, které budou při editaci zrušené, nebo maximální počet l po sobě jdoucích vět, které budou přidané. Počet vkládaných vět v prvním případě a rušených vět v druhém případě omezený není. Program B je ovšem pro obě varianty rozdílný.

Podívejme se např. na variantu programu B omezujícího počet po sobě jdoucích rušených vět na číslo k. Program vyhradí ve vnitřní paměti prostor pro uchování k+1 vět souboru S. Činnost programu je založena na následujícím tvrzení: Jestliže větu souboru N nelze spárovat se žádnou z k+1 po sobě jdoucích vět souboru S, jde o větu v průběhu editace do souboru přidanou.

Program B u neměněných vět registruje pouze jejich počet, zrušené a přidané věty vypisuje do protokolu s udáním jejich pořadového čísla. Změněné věty vypisuje ve tvaru řádek starých hodnot, řádek nových hodnot. Pokud v celém řádku nedošlo ke změně, řádek nových hodnot je vyplněn rovnítky. Z důvodů jednoduchosti program B porovnává úseky vět souborů S a N o délce jednoho řádku obrazovky terminálu. Výpisy vět mají rovněž tvar, v jakém byly zobrazeny na obrazovce v průběhu editace. Součástí protokolu je

statistika počtu rušených, přidaných a měněných vět. Program B spouštíme z terminálu a také první výstup protokolu směrujeme na obrazovku. Pokud výsledek editace není uspokojivý, znovu vyvoláme textový editor, tentokrát na soubor N.

Pokud jde o závislost programu B na konkrétní aplikaci, je nutno brát v úvahu maximální počet k resp. l rušených resp. přidávaných vět, délku věty souborů S a N, délku klíče a pozici položek klíč tvořících. Protože délku věty a pozice klíčových položek určuje program A, lze jediným programem B zabezpečit více aplikací. Běžnou praxí např. je, že doplněním mezer natáhneme délku věty na násobek délky řádku obrazovky.

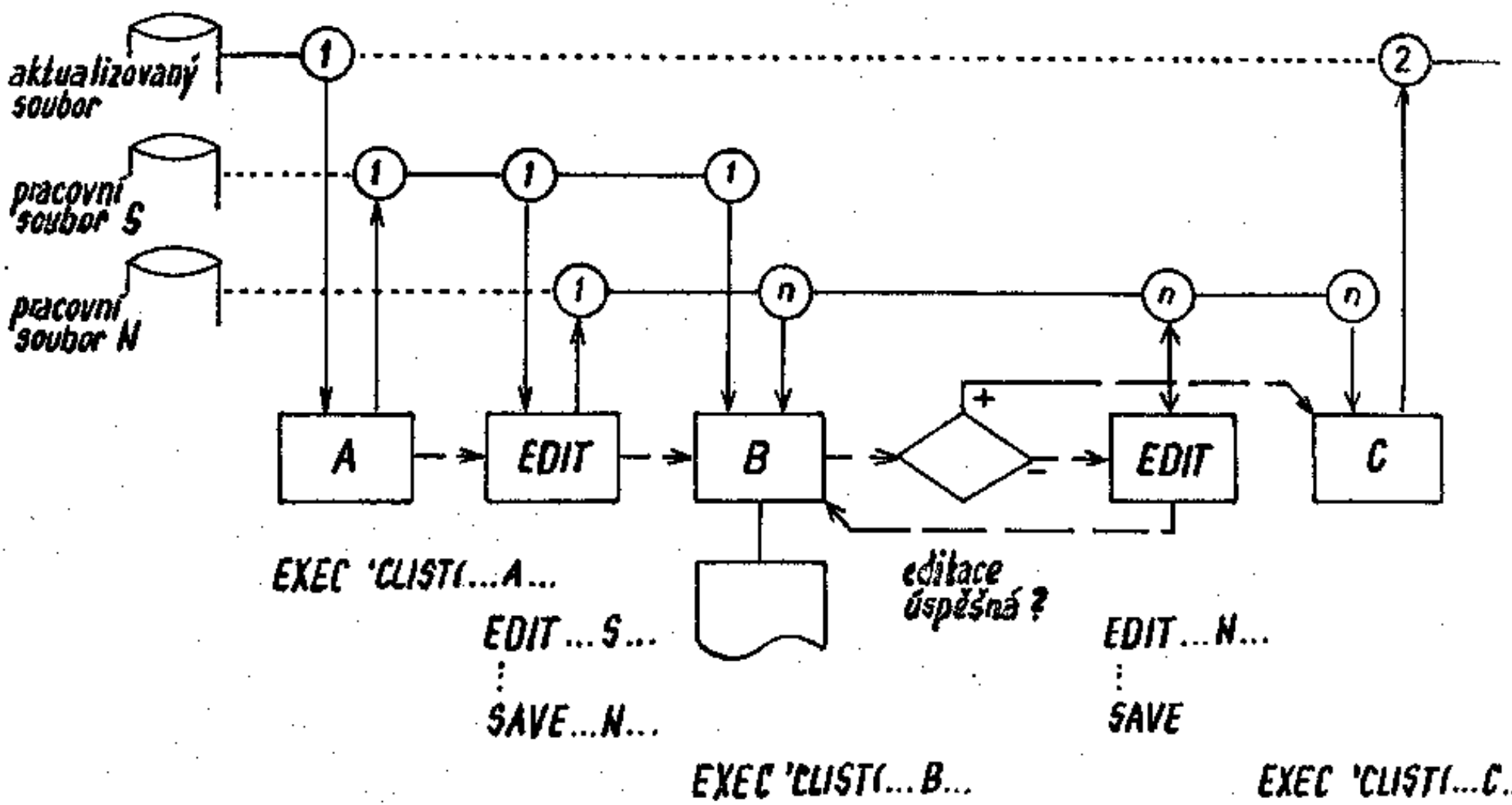
Na závěr je nutné protokol vytisknout jako písemný doklad o provedených změnách.

5. Návrat k aktualizovanému souboru

Pokud editovaný soubor byl souborem aktualizovaným, je nutné např. přejmenováním ztotožnit aktualizovaný soubor se souborem N. Byl-li však použit program A, je třeba zajistit přepis souboru N do aktualizovaného souboru. Tuto činnost zajistí další na aplikaci závislý program (označovaný dále C). Program C je ověem snadno odvoditelný z programu A záměnou vstupního a výstupního souboru a otočením přiřazovacího příkazu na tvar U=V, BY NAME;

6. Závěr

Návaznost jednotlivých kroků aktualizace ukazuje obr. 1. Činnost u terminálu vykonává provozní programátor. Obvyklé však je, že se terminálové relace zúčastní uživatel, který řídí provádění změn a schvaluje závěrečný protokol.



Obr. 1 Aktualizace datového souboru