Ing. Jaroslav K l e č k a
OKR Automatizace řízení, k.ú.o.

VZTAH DVOJICE PROGRAMÁTOR - OPERÁTOR

Na loňském semináři došlo k rozdělení programátorů do dvou kategorií, a to na p r o g r a m á t o r y a na P r o g r a m á t o r y . Ten, kdo tento seminář absolvoval, si možná vzpomene, že zmíněné dělení se zde různým způsobem obměňovalo, při čemž původní definice vznikla na základě toho, zda příslušný programátor pracuje v nižším či vyšším programovacím jazyce.

Já se však domnívám, že dělení lze znovu sjednotit a to na základě následujícího tvrzení. Téměř každý programátor se snad již z principu brání tvrdit totéž, co klasici ..... V í m , ž e n i c n e v í m ..... a je přesvědčen, že ví vše nebo alespoň téměř vše a myslí si, že je p a n P r o g r a m á t o r.

Co mne přivedlo k tomuto tvrzení? Měl jsem možnost si poslechnout magnetofonový záznam loňského semináře a nabyl jsem dojem, že programátor je člověk neomylný, který by velice rád všem kolem sebe diktoval, jak mají pracovat, aby mu jeho vlastní práci co nejvíce usnadnili. Týká se to nejen systémových analytiků, ale i provozních pracovníků. Co by však měl udělat on sám pro usnadnění práce všech ostatních kolem sebe, o to se již moc nezajímá.

Pracuji ve výpočetní technice již dosti dlouho na to, abych poznal práci jak na počítačích bez vyššího programovacího jazyka, tak s ním. Prakticky vždy jsem se však setkal se zajímavou vlastností našeho jednicového P r o g r a m á t o r a .

Vždy a za všech okolností se snažil udělat nepostradatelným a pokud možno, znepříjemňoval život sám sobě, ale především pak také jednicovému O p e r á t o r o v i . Mezi uvedenými dvěma veličinami panuje zřejmě odvěký boj, trvající snad již od doh rozdělení těchto dvou funkcí a vítězství v tomto boji se kloní tu na jednu, tu na druhou stranu.

Jaké metody ve zmíněném boji používá Programátor. Ať už je provoz počítače /pochopitelně III. generace, pracujícího v režimu multizpracování a instalovaného s tím záměrem, aby toho co nejvíce zpracoval/ organizován jako uzavřený /closed shop/ nebo není, snaží se pan Programátor k tomuto počítači za každou cenu alespoň někdy podívat a zjistit, jak se tam jeho obsluha vede. Je samozřejmé, že celá řada úloh obecného charakteru, jejichž ladění je složité a spolupracuje s operačními systémy, si většinou vyžaduje přítomnost alespoň systémového programátora.

Je však s podivem, že téměř každý pravý Programátor se větší či menší část svého života domnívá, že každý jeho program má tento výjimečný charakter a je přesvědčen, že pro svoji další práci musí bezpodmínečně vidět, zda se mu točí magnetické pásky nebo čtou děrné štítky. Čím déle ve svém oboru pracuje, tím je tato snaha menší, ale vždy a za všech okolností se domnívá, že při této své přítomnosti u počítače se mu budou věnovat nejméně dva operátoři. Tito operátoři musí okamžitě reagovat na každé přání našeho Programátora. Ten si však většinou neuvědomí, že tito operátoři by měli současně obsloužit ještě nějaký ten další program a využili tak vlastností počítače III. generace.

Dejme tomu, že provoz počítače uzavřeme a Programátoro-

vi přístup zamezíme. Programátor se však nevzdává. Určitě se pokusí udělat se nepostradatelným. Jak toho dosáhne? Začně ladit svůj "výjimečný" program a pošle jej na zkušenou do světa. Na cestu mu toho s sebou moc nedá a podle toho se mu také program vrátí zpět. Co má na příklad operátor udělat s programem, který začne cyklovat. Nejjednodušší z celé řady druhů cyklů programu je ten, kdy program najednou přestane pracovat s periferními jednotkami a pokud možno, zastaví činnost všech ostatních programů. Daleko horší však už je případ, kdy program pracuje třeba s tiskárnou a na této jednotce vytiskne třeba i 1500 listů papíru, na kterých jsou navíc úplné nesmysly nebo tímtéž začne popisovat už šestou výstupní cívku magnetické pásky. Velice výhodné cykly jsou také ty, kdy se program nějakým záhadným způsobem sám ukončí a uvedené nesmysly se třeba separují a po obou stranách se ořeže vodící děrování, takže tento papír již nejde ani obrátit a potisknout z druhé strany něčím rozumějším. Takovéto případy se většinou stávají tehdy, když se náš pan Programátor rozhodne použít ladící data z rutinních rozsahů a své ladění žádným způsobem neomezí. Operátor si takovýchto geniálních výstupů občas všimne včas a program vymaže. Pokud však Programátorovo dítko na svoji cestu do světa nedostalo patřičné náležitosti, operátor udělá jen to, co ze svých chatrných zkušeností zná, a to je, že program nějakým způsobem vypíše a k tomu vytiskne kousek vstupních souborů. Udělá-li to však z své vlastní újmě, určitě to nebude tak, jak to Programátor pro odstranění chyby bude potřebovat.

Není přece nic jednoduššího, než popsat normální činnost programu, odhadnout čas zpracování a popsat operátorovi činnost, kterou by měl vykonat při abnormálním konci programu. Tento popis činnosti lze udělat jak na obyčejném kusu papíru, tak na zavedeném požadavkovém listu či jiném formuláři. Ještě jednodušší situace nastává při použití operačního systému, kdy si chod programu vlastně řídí sám programátor pomocí řídících výroků řídícího jazyka operačního systému. Programátorovo dílo je však většinou tak geniální, že nic tako-

vého nepotřebuje ani po "úplném odladění" a stačí pak, aby se program dostal jednou za deset let do větve, kterou ještě nikdy nešel a světe div se, ta o v i č e n á o p i c e /jak se také někdy operátor nazývá/, neví kudy kam a musí třeba za 7 minut půl třetí ráno zvednout telefon a našeho programátora k jeho dílu pozvat. Když to náhodou neudělá a výjimečné stavy ošetří dle své dobré vůle, končí všechny ty výpisy o nějaký ten řádek dříve, než bude zapotřebí.

První reakcí Programátora asi nebude hledání vlastní chyby, ale tvrzení, kdyby byl býval u toho byl, nebylo by se nic stalo a vše by na místě odstranil. Vidíte, už zase se mu zachtělo k počítači.

Řekli jsme si tedy, že program by měl mít na své pouti k dispozici celou řadu náležitostí, aby mohl samostatně žít. Sem také patří odhad času zpracování. Jak je tento odhad důležitý při ladění, jsme si již řekli. Programátor se však nezřídka uchyluje k druhému extrému a délku zpracování podhodnotí. Každý operátor však již zná tohoto specialistu a když po trojnásobku uvedeného času program vyasaže, přijde určitě příští den někdo na to, že program by byl býval za 20 sekund skončil. Kde teď ale vzít ty tři hodiny na celé konfiguraci, co tento program bude muset znovu pracovat: Opět pádný důvod, proč u toho být.

Existuje dobrý způsob, jak operátorovi jeho práci usnadnit. Je jím zavedení samovysvětlujících zpráv, vypisovaných při chodu programu na ovládacím psacím stroji. Program si v podstatě sám řekne, jak se s ním má zacházet. I zde je však nutné postupovat uvážlivě a pokud možno nepsat na stroji o rychlosti 10 znaků/sec. zprávy o rozsazích 50 a více znaků. Zde je pak nutné volit krátké standardní zprávy. Ze svých zkušeností však vím, že nezavede-li se tento standard kódovaných zpráv dostatečně brzo po instalaci počítače, naučí se je Programátor, jako člověk značně konservativní, používat o hodně později než operátor.

Současně se bude určitě snažit zavést svoje vlastní a jen pro něho srozumitelné zkratky. Stejné metody pak Programátor používá při tvorbě symbolických jmen parametrů procedur, sestavených z výroků řídícího jazyka operačního systému. Pro zjednodušení své /ne už cizí/ práce si tato symbolická jména zvolí úplně odlišná od standardu, na který je operátor zvyklý. Ten se potom celou řadu minut snaží zjistit, jak proceduru modifikovat, aby celou práci mohl provést za změněných provozních podmínek, jakými je třeba porucha některého z používaných zařízení. Pamatujme, že dnešní operační systémy předpokládají flexibilitu zpracování.

Kromě dokazování své nepostradatelnosti se náš Programátor pokouší znepříjemnit operátorovi jeho práci i jinak. Zmínil jsem se již o řídícím jazyku operačního systému. Je to vlastně další programovací jazyk, který se musí naučit nejen Programátor. I když jej tento Programátor bude znát již dostatečně dlouho, bude v něm vždy nějakou tu chybu dělat. Běda však operátorovi, který se jej nenaučí včas a lépe než Programátor a nebude jeho chyby opravovat. Vždyť ten operátor našeho Programátora v jeho rozletu brzdí! Je asi velice obtížné si uvědomit, že počty programů či programových chodů, zpracovávaných denně na dnešních počítačích se pohybují ve stovkách a z nich i několik desítek má nějakou chybu v řídících výrocích. Některé anomálie jsou dokonce tak zajímavé, že se do nich ani operátorovi nechce.

Řekli jsme si už také, že systém, pracující s multizpracováním, má být co nejdokonaleji využíván, aby jím za den prošlo co největší množství dat. Znamená to tedy, že operátor musí jednotlivé programy či programové chody mixovat tak, aby celá konfigurace počítače byla využita co nejefektivněji. Náš Programátor si ale myslí, že operátor by měl být věsvědomí, aby poznal, co je mu ke zpracování předkládáno a co si program během svého chodu vyžádá. Nejen, že nenapíše kolik a jakých periferních jednotek či velikostí pracovních prostorů na diskových jednotkách bude

program vyžadovat, ale nezřídka ani nenapíše sériová čísla požadovaných paměťových nosičů dat anebo a co je ještě horší napíše tato čísla úplně jiná. Operátor pak někdy z čirého zoufalství sáhne k té nejtěžší ze svých zbraní v odvěkém boji s Programátorem a tou je vrácení zakázky, aniž ji začal vůbec zpracovávat. Boj se pochopitelně dále přiostřuje, protože v tomto stádiu do něj zřejmě začnou zasahovat vedoucí pracovníci a slabší se stáhne do předem připravených pozic. Bohužel to velice často bývá právě Programátor, kdo vítězí, protože musí splnit "pekelné" termíny.

Pokusme se tento boj alespoň částečně zmírnit a stanovme si nějaká pravidla hry, při níž by obě strany zůstaly celé a jejímž výsledkem by byl co největší objem vykonané práce. Hlavními zásadami by tedy měly být následující body:

1. Zásadně dodržovat uzavřený systém práce na sále počítače /closed shop/ - kvalitní operátor má dostatečný klid a prostor pro přípravu a obsluhu všech programů při jejich zpracování.

2. Při stavbě programů dodržovat perfektně všechny zavedené standardy, umožňující co nejefektivnější zpracování a dosažení vysoké průchodnosti dat počítačem.

3. Vypracovat a dodržovat standardy pro vyplňování pokynů zpracování všech programů či programových chodů. Sem patří i standardy pro tvorbu procedur, sestavovaných z výroků řídícího jazyka používaného operačního systému.

4. Perfektně vyplňovat všechny požadavky programů či programových chodů na zdroje počítače a umožnit tak vhodné sestavování vstupních front zakázek. Současně je třeba vždy vyplňovat správná sériová čísla magnetických paměťových médií či správný počet požadovaných kopií výstupních sestav.

Všechny standardy je nutno zavést v dostatečně krátké

době po zahájení práce na počítači, ne-li ještě před tímto zahájením. Neuděláme-li to totiž včas, může se nám stát, že již nikdy nepodchytíme správný vztah dvojice P r o g r a m á t o r - O p e r á t o r a běda pak každému vedoucímu pracovníkovi, který se o to později pokusí.

Jestliže jsem úvodem vzpomněl loňského semináře, zakončím svůj příspěvek ve stejném duchu, tedy vzpomínkou: Tehdy totiž bylo velice pravděpodobné, že někdo někoho ve svém referátu či diskusním příspěvku napadl. Já však doufám, že jsem tímto napadl všechny programátory, a to zcela záměrně.