ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1975

Seria: Automatyka z. 34 Nr kol. 458

Maciej Bargielski,
Janusz Czapnik

INSTYTUT KOMPLEKSOWYCH SYSTEMÓW STEROWANIA

PROBLEMY KONSTRUKCJI BIBLIOTEKI WIELODOSTĘPNYCH PROCEDUR SYSTEMOWYCH

Streszczenie. Dla usprawnienia korzystania z maszyny cyfrowej, w skład jej oprogramowania podstawowego wchodzi biblioteka procedur standardowych. Praca niniejsza omawia pewne problemy związane z konstrukcją podprogramów wchodzących w skład tej biblioteki dla wielodostępnego systemu obliczeniowego. Omówiono tu różne sposoby zapewnienia wielodostępności procedur - zarówno w oparciu o stos systemowy, czy o stos własny, biblioteki, jak i powielanie treści procedury.

1. Wstęp

Przez bibliotekę procedur systemowych rozumie się zbiór procedur rozwiązujących typowe problemy użytkownika, przygotowany przez konstruktorów systemu cyfrowego. Procedura systemowa musi być tak skonstruowana, by użytkownik mógł dołączać ją dowolną ilość razy do pisanego przez siebie programu bez konieczności przepisywania treści procedury, a jedynie przez jej wywołanie. Biblioteka procedur stanowi więc istotną część oprogramowania cyfrowego systemu obliczeniowego a jej skład decyduje o możliwościach upraszczania i skracania treści programów użytkownika.

Jeśli w systemie cyfrowym nie ma możliwości jednoczesnego wykonywania kilku programów, zarówno konstrukcja jak i rozbudowa biblioteki procedur nie przedstawia większych trudności. Sytuacja znacznie się komplikuje,gdy system realizuje jednocześnie kilka programów lub program liczony może być w dowolnej chwili przerwany przez program o wyższym priorytecie. W dalszych rozważaniach przedstawimy kilka możliwych metod rozwiązania tego zagadnienia.

2. ORGANIZACJA PROCEDUR W SYSTEMIE JEDNOPROGRAMOWYM

Przy konstruowaniu procedur systemowych istotna jest odpowiedź na następujące pytania.:

1. Jak procedura będzie wywoływana?
2. Jak będą do procedur przekazywane argumenty?
3. Gdzie będą przechowywane wyniki pośrednie?
4. Gdzie procedura prześle wyniki końcowe?

W systemie jednoprogramowym wszystkie te problemy mogą być rozwiązane stosunkowo prosto. Wywołanie procedury może nastąpić bądź rozkazem "skoku ze śladem", który spowoduje wpisanie aktualnego stanu licznika rozkazów do obszaru treści procedury, bądź przed wywołaniem zapisać adres powrotu w rejestrze indeksowym. W obu wypadkach procedura musi się kończyć rozkazem skoku do miejsca wskazanego przez adres powrotu.

Dla ułatwienia użytkownikowi korzystania z procedur systemowych są one z reguły wywoływane przez podanie symbolicznej nazwy procedury. Nazwa ta jest w czasie translacji identyfikowana przez assembler i zastępowana fizycznym adresem procedury.

Przekazywanie do procedury wartości argumentów może być zorganizowane w różnoraki sposób. W przypadku niewielkiej liczby argumentów ich wartości mogą być w momencie wywołania procedury umieszczone w rejestrach procesora. Gdy to jest niemożliwe, można zażądać, by przed wywołaniem wartości argumentów zostały przesłane do określonych adresów pamięci operacyjnej (z reguły adresom tym przyznaje się stałą nazwę symboliczną, zastępowaną przy translacji adresem fizycznym). Inne możliwości to:

- użytkownik umieszcza wartości argumentów w swoim programie-ich rozmieszczenie jest zdeterminowane względem rozkazu wywołania procedury,
- użytkownik przekazuje do procedury adres początku bloku danych (ewentualnie również długość bloku).

Tak skonstruowana biblioteka procedur systemowych może pracować poprawnie, gdyż praca jednoprogramowa zapewnia, że podczas wykonywania procedury stan pamięci jest jednoznacznie określony działaniem samej procedury.

3. PROCEDURY WIELODOSTĘPNE

Współczesne cyfrowe systemy obliczeniowe dla zwiększenia efektywności obliczeń z reguły pracują w reżimie wieloprogramowym (tzn. w danym odcinku czasu może być jednocześnie wykonywanych kilka programów). W systemach czysto obliczeniowych stosuje się zwykle podział czasu między równolegle wykonywane programy. W systemach cyfrowych sterujących procesami przemysłowymi wybór programu aktualnie liczonego jest uwarunkowany sytuacją zewnętrzną i program liczony może być w dowolnej chwili przerwany przez program o wyższym priorytecie.

Praca wieloprogramowa narzuca szczególne wymagania na konstrukcję biblioteki procedur systemowych. Najważniejszym postulatem jest wielodostępność - dowolna procedura biblioteczna musi być dostępna dla wywołania przez każdy z jednocześnie realizowanych programów. Powstają więc następujące problemy:

1. Należy zapewnić, by przy jednoczesnym wywołaniu procedury przez kilka programów każdorazowo prawidłowo odtworzyć adres powrotu. Ponieważ wewnątrz procedur mogą się znajdować odwołania do innych procedur, odtworzenie adresu powrotu do programu użytkownika może wymagać zapamiętania kilku adresów pośrednich.

2. Przy każdym wywołaniu procedura powinna operować na autonomicznym zbiorze argumentów. Ponowne wywołanie procedury nie powinno wpływać na wyniki pośrednie, obliczone przez procedurę w programie przerwanym.

Przedstawione wymagania sprawiają, że dla procedury systemowej w warunkach pracy wieloprogramowej przechowywanie w obszarze procedury adresu powrotu i argumentów bieżących jest niedopuszczalne. Informacja ta nie może być również przechowywana w rejestrach jednostki arytmetyczno-logicznej.

Należy tu dla ścisłości zaznaczyć, że dla dowolnej konstrukcji biblioteki procedur systemowych można zapewnić wielodostępność procedur, jeśli zapewni się blokadę możliwości przerwania liczenia programu na czas wykonywania procedury. Czas wykonywania procedury może być jednak długi (procedura może np. wymagać wykonania operacji wejścia/wyjścia), więc takie rozwiązanie prowadzi z reguły do niedopuszczalnego wydłużenia czasu reakcji systemu.

4. ORGANIZACJA PROCEDUR W SYSTEMIE WIELODOSTĘPNYM

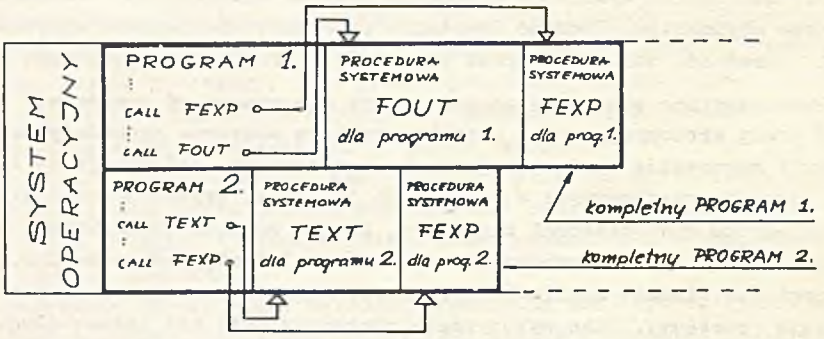
Procedury wielodostępne, spełniające wymagania sformułowane w poprzednim punkcie, mogą być realizowane w różnorodny sposób. Wybór jednego z nich uzależniony jest od konfiguracji urządzeniowej i przeznaczenia systemu i dlatego nie można zdecydować a priori o wyższości jednego sposobu nad innym. Rozpatrzymy tu organizację procedur wielodostępnych przez powielenie, w oparciu o stos systemowy i w oparciu o stos własny.

4.1. Powielanie procedur

Powielanie treści procedury jest najprostszym sposobem uzyskiwania wielodostępności. Nie nakłada on na konstrukcję (organizację) takiej procedury żadnych wymagań dodatkowych, poza oczywiście budową typową dla dowolnej procedury w danym systemie. Idea ta polega na zwielokrotnieniu ilości procedur wspólnych tak, aby każdy program użytkowy miał swoje "prywatne" procedury, z których tylko on może korzystać.

W systemie tym kompletny program użytkowy tworzony jest przez system operacyjny w czasie wprowadzania głównej części tego programu do pamięci. W tym czasie system operacyjny notuje wszystkie odwołania do procedur standardowych, przydzielając im równocześnie nowe adresy, zlokalizowane zwykle w obszarze spójnym z wprowadzonym programem. Po zakończeniu wprowadzania tej części programu, następuje przekopiowanie z biblioteki systemowej tych wszystkich procedur standardowych, których dany program używał. W efekcie otrzymuje się pełny blok programowy, zawierający wszystkie potrzebne procedury.

Wprowadzanie drugiego programu powoduje podobne działanie systemu operacyjnego. W efekcie może się zdarzyć (z dużym prawdopodobieństwem), że pewne procedury będą znajdowały się w innych miejscach pamięci, będąc przydzielone innym programem użytkowym. Przykładową mapę pamięci w przypadku dwóch programów przedstawia rys. 1.



Rys. 1. Mapa pamięci dla dwóch programów w systemie z powielaniem procedur

Zasadniczą wadą przedstawionego tu sposobu uzyskiwania wielodostępności procedur systemowych jest nieoptymalne wykorzystanie pamięci, przejawiające się w tym, że pewne podprogramy występują wielokrotnie w różnych miejscach pamięci, co powoduje poniekąd zatracenie celu stosowania podprogramów. Jednak zalety tego rozwiązania, jakimi są:

- prostota organizacji procedur,
- skrócenie czasu obliczeń

powodują, że rozwiązania takie są często spotykane, szczególnie w systemach z zewnętrznymi pamięciami masowymi, których ograniczenie pojemności pamięci dostępnej dla programów użytkowych nie jest tak drastyczne.

4.2. Procedury wielodostępne pracujące w oparciu o stos systemowy

Wszystkie programy pracujące w systemie czasu rzeczywistego mogą być przerywane w wyniku zaistnienia jakiejś określonej sytuacji zewnętrznej. Po zaniknięciu tej sytuacji przerwany program powinien być liczony od miejsca, w którym przerwanie nastąpiło. Aby umożliwić kontynuację programu od danego miejsca, w maszynie liczącej muszą być zachowane i odtworzone wszystkie warunki limitujące pracę programu. Do warunków tych należą:

- stan licznika operacji, pamiętający adres aktualnie wykonywanego (czy następnego) rozkazu,
- stan wszystkich rejestrów,
- stan komórek roboczych programu.

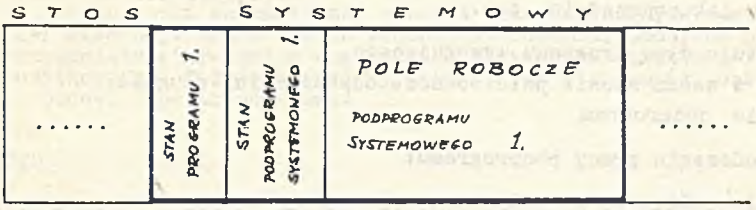
System operacyjny, pod nadzorem którego pracuje maszyna, spełnia to zadanie w odniesieniu do dwóch pierwszych warunków, przechowując je w spe-

cjalnym zbiorze uporządkowanym i jednostronnie dostępnym, zwanym stosem. Komórki robocze programu nie są w nim chronione; muszą więc one być tak zlokalizowane, aby praca innego programu ich nie zniszczyła.

W przypadku wywołania podprogramu z biblioteki systemowej sytuacja zmienia się. Każdy z tych podprogramów może być w dowolnej chwili wywołany i dlatego w stosie systemu operacyjnego muszą być dodatkowo przechowywane komórki robocze tych programów. Przerwanie programu w trakcie jego odwoływania się do procedury bibliotecznej powoduje, że w stosie znajduje się:

- stan programu,
- stan podprogramu bibliotecznego,
- komórki robocze podprogramu bibliotecznego,

jak pokazano przykładowo na rys. 2.



Rys. 2. Przykładowy element stosu systemowego

Rozpoczęcie nowego programu nie niszczy więc warunków potrzebnych do odtworzenia przerwanego programu, nawet w przypadku wywołania przez nowy program tego samego podprogramu systemowego.

Ten sposób organizacji procedur wielodostępnych nakłada dodatkowe wymagania na system operacyjny i na konstrukcję samej procedury. Konstrukcję tę można przedstawić wg następującego schematu:

- zapisanie w stosie systemowym śladu programu (LO),
- otwarcie w stosie pola roboczego o potrzebnej długości,
- realizacja programu w oparciu o dane z pola roboczego w stosie,
- zamknięcie (likwidacja) pola roboczego w stosie,
- powrót do programu.

System operacyjny musi więc mieć możliwość wydłużania i skracania stosu o dowolną, żądaną przez programy wielkość wraz z informacją o początku przyznanego obszaru.

Przedstawiony tu sposób organizacji procedur wielodostępnych w oparciu o stos systemowy posiada zarówno zalety, jak i wady. Do niewątpliwych zalet należy oszczędność pamięci na przechowywanie programów, gdyż treść programu znajduje się w pamięci tylko w jednym miejscu, zaś pól roboczych jest co najwyżej tyle, ile aktualnie liczonych programów. Do wad zaliczyć trzeba większy stopień komplikacji treści procedur (często większa dłu-

gość) jak i, na ogół, zwiększenie czasu liczenia. Również system operacyjny musi być bardziej skomplikowany, co nie jest bez wpływu zarówno na zajętość pamięci, jak i czas reakcji systemu.

4.3. Procedury wielodostępne pracujące w oparciu o własny stos

Czynności systemu operacyjnego, obsługujące procedury biblioteczne, mogą być przyjęte przez miniprogram zarządzający obsługą procedur. W tym przypadku stos systemowy zawiera jedynie ślady przerwanych programów, zaś ślady programów wywołujących procedury wielodostępne oraz pola robocze tych procedur znajdują się w innym stosie - stosie własnym procedur wielodostępnych.

Wywołanie każdej procedury polega wtedy na odwołaniu się do programu zarządzającego z odpowiednimi parametrami, określającymi bliżej parametry wołania. Program zarządzający uruchamia odpowiedni podprogram w sposób analogiczny jak poprzednio, tzn.:

- zapamiętuje ślad programu wywołującego,
- otwiera w swoim stosie pole robocze odpowiedniej długości,
- uruchamia podprogram

Po zakończeniu pracy podprogramu:

- likwiduje pole robocze podprogramu,
- przekazuje wyniki do programu wołającego,
- uruchamia dalszy ciąg programu wołającego.

Jak widać następuje tu rozdział zadań systemu operacyjnego odnośnie obsługi programów użytkowych i obsługi wielodostępnych procedur bibliotecznych. Posiada to istotny aspekt praktyczny, gdyż okazuje się, że jedynie obsługa procedur wymaga rezerwowania w stosie pola roboczego o zmiennej długości. W przypadku obsługi zaleceń programów użytkowych lub obsługi przerwań zewnętrznych elementy stosu mogą mieć stałą długość. Dekompozycja pozwala więc na znaczne uproszczenie części podstawowej systemu operacyjnego i pozwala zmniejszyć czas reakcji systemu zarówno na zlecenia programowe, jak na przerwania zewnętrzne. Płaci się jednak za to zwiększeniem obszaru pamięciu zajmowanego przez system, gdyż podwójny stos na ogół wymaga zarezerwowania większego obszaru niż stos uniwersalny.

5. Wnioski końcowe

Przedstawione wyżej rozważania szkicują możliwości konstrukcji wielodostępnych procedur systemowych. Wybór jednej z nich wymaga szczegółowej analizy własności urządzeniowych systemu. Przykładowo, w systemie z pojemnością pamięcią masową o stosunkowo krótkim czasie dostępu, obszar pamięci operacyjnej zajmowany przez program przestaje być parametrem krytycznym. Dla zapewnienia maksymalnej szybkości pracy systemu można w tym przypadku przewidzieć powielanie procedur.

Z kolei dla systemu wyposażonego jedynie w pamięć operacyjną o niewielkiej pojemności niezbędne staje się rozwiązanie w oparciu o stos. Wybór między stosem "systemowym" a autonomicznym monitorem obsługi procedur należy przeprowadzić, biorąc pod uwagę stopień komplikacji systemu operacyjnego, zajmowany przez niego obszar pamięci oraz czas obsługi zleceń programowych w każdym przypadku.

Należy podkreślić, że przedstawione tu rozwiązania nie wykluczają się wzajemnie. Analiza czasowa wykazuje, że optymalne jest z reguły właśnie rozwiązanie hybrydowe. W rozwiązaniu tym część procedur o stosunkowo krótkim czasie wykonywania konstruuje się jako nieprzerywalne, inne - z możliwością przerywania wg wyżej podanych metod.

LITERATURA

[1] Donovan J.J.: Systems Programming". Mc Graw-Hill 1972..
[2] Fisher F.P., Swidnle G.F. "Computer Programming Systems". Holt. Rinehart and Winston Inc. 1964.
[3] Flores I.: "Computer Software". Prentice-Hall 1965.
[4] Martin J.T.: "Real-Time Computer Systems". Prentice-Hall 1967.
[5] PDP-10 Time-Sharing Handbook".Digital Equipment Corp. 1970.
[6] "Real-Time Executive Software System". Hewlett-Packard Comp.1971.
[7] "Varian 620/F,L Computer Handbook". Varian Data Machines 1971

ПРОБЛЕМЫ МНОГОВОСТУПНЫХ СТАНДАРТНИХ ИРОЦЕДУР

Р е з ю м е

Б работе представлены некоторе проблемы, связанне с конструированием подпрограмм входящих в состав библиотеки многодоступной вычислительной системы.

Описываются разные способы реализации многодоступности - и так, опираясь на стог операционной системы, или на собственный стог библиотеки, или, наконец, на дуплицирование тела процедуры.

PROBLEMS COUCERNING THE CONSTRUCTION OF RE-ENTRY SYSTEM LIBRARY

S u m m a r y

In this paper some problems concerning the construction of a multi-access procedures library are presented.Various methods of this are shown e.g. using an operating system stack, an own library stack as well as the duplicating of a procedure body.