mgr inż. JACEK JURKOWSKI

Przemysłowy Instytut Automatyki
i Pomiarów MERA-PIAP

Warszawa

O EFEKTYWNYM PISANIU PROGRAMÓW SPECJALNYCH

W artykule zwrócono uwagę na diagnostykę błędów występujących podczas uruchamiania programów specjalnych pod kontrolą systemów operacyjnych układów automatyki kompleksowej. Zaproponowano postępowanie ułatwiające uruchamianie programów specjalnych w warunkach ograniczonej diagnostyki błędów wykonania.

1. Wstęp

Opracowane dotychczas w MERA-PIAP systemy operacyjne (oprogramowanie podstawowe) układów automatyki kompleksowej, takie jak: SZPAK, SORT, ABAK, pozwalają na uruchamianie pod ich kontrolą programów specjalnych.

Programy specjalne są stosunkowo dużymi (ok. 6k) programami, pisanymi zwykle w Fortranie. W zrealizowanych systemach ich uruchomienie nie wiąże się z żadnym konkretnym momentem czasu, ani z obsługą zmiennych procesu, chociaż programy te korzystają z wartości zmiennych procesu. Umożliwiają one wykonywanie dużych i złożonych obliczeń wykorzystujących dane zczytywane z instalacji technologicznej.

Programy specjalne mogą komunikować się ze zmiennymi procesu tylko poprzez przeznaczone do tego celu systemowe podprogramy. Podobnie, komunikacja z urządzeniami wejścia wyjścia jest możliwa tylko poprzez system operacyjny. Ten ostatni czynnik, w połączeniu z koniecznością *oszczędzania pamięci* powoduje, że uruchamiane (w sensie ich weryfikacji) programy specjalne mają bardzo ubogą diagnostykę błędów wykonania lub nie mają jej wcale, wyjątkiem jest system operacyjny SZPAK [3]. Jest to bardzo niewygodne dla użytkownika układu automatyki kompleksowej, który przecież nie jest zawodowym programistą, lecz inżynierem chemikiem czy cukrownikiem. Użytkownik powinien sam ulepszać oprogramowanie użytkowe układu, w ramach możliwości jakie daje mu system operacyjny. Wydaje się, że w niektórych systemach możliwości te dotyczące programów specjalnych są ograniczone, co może przy pierwszym zetknięciu się z systemem (najczęściej właśnie przez programy specjalne) zniechęcić użytkownika do układu jako całości.

Poniżej omówiono kilka problemów związanych z wykrywaniem i usuwaniem błędów w programach specjalnych, które mają pracować pod kontrolą wymienionych systemów operacyjnych, zrealizowanych w maszynie cyfrowej Odra 1325.

2. Wykrywanie błędów wykonania w programach uruchamianych off-line i w programach specjalnych

Uruchamiając off line (tylko pod kontrolą programu zarządzającego Executive) program napisany w Fortranie, programista ma do dyspozycji trzy poziomy wykrywania błędów wykonania (poziomy śledzenia). Poziom śledzenia deklaruje się w wierszach sterujących kompilacją, na jeden z trzech

poniższych sposobów:

TRACE 2
TRACE 1
TRACE 0

Najwyższym poziomem śledzenia jest TRACE 2. Po wykryciu błędu wykonania podprogram śledzący wyprowadza przez urządzenie zewnętrzne (np. drukarkę wierszową) kod błędu, odpowiednio zredagowane informacje o 100 ostatnio wykonanych instrukcjach, pod nagłówkiem STATEMENT TRACE, oraz informacje o ostatnich 25 wejściach i wyjściach z segmentów programu, pod nagłówkiem SEGMENT TRACE. Poziom TRACE 1 zapewnia nieco uboższą diagnostykę, nie jest wykonywany STATEMENT TRACE. Na poziomie TRACE 0 jest wyprowadzany tylko kod błędu wykonania. Informacje dotyczące błędu są wyprowadzane przez urządzenie wyjściowe, zadeklarowane w pierwszym napotkanym przez kompilator wierszu sterującym OUTPUT.

Zastosowanie danego poziomu śledzenia wiąże się ze zwiększeniem obszaru pamięci operacyjnej, jaki zajmuje program, a także z wydłużeniem czasu wykonania programu. Im wyższy poziom śledzenia zastosujemy, tym więcej pamięci zajmie program i tym dłużej będzie się on wykonywał. Ogólnie zaleca się, aby podczas testowania programu stosować poziom TRACE 2, w trakcie normalnej eksploatacji poziom TRACE 1, a jeśli program jest dokładnie sprawdzony i istotne jest ograniczenie wielkości programu oraz czasu jego wykonywania, wówczas można zastosować poziom TRACE 0.

W przypadku programów napisanych w Planie, programista ma do dyspozycji podprogramy śledzące typu MONITORA i TRACEA [5].

Najczęściej, uruchamiając programy specjalne, nie można korzystać z wymienionych powyżej udogodnień, ponieważ istnieje górna granica wielkości programu specjalnego, narzucona przez oprogramowanie podstawowe. Program właściwy (część wykonująca obliczenia) może być tak duży, że na procedury śledzenia odpowiednio wysokiego poziomu nie ma już miejsca w pamięci operacyjnej maszyny. Drugą przyczyną, również narzuconą przez oprogramowanie podstawowe jest pośrednia, przez system operacyjny, komunikacja z urządzeniami wejścia/wyjścia. Wyklucza to deklarowanie urządzeń zewnętrznych w wierszach sterujących kompilacją, co w przypadku programów napisanych w Fortranie nie pozwala procedurom śledzącym na wyprowadzanie informacji o błędzie.

Pomocnicze podprogramy, stosowane w programach napisanych w Planie, nie mogą być użyte ze względu na inny tryb współpracy z urządzeniami wejścia/wyjścia (oprogramowanie podstawowe współpracuje z nimi, zwykle, w trybie DRM).

Uruchamianie programów off-line jest znacznie łatwiejsze od uruchamiania programów specjalnych. Poniżej zaproponowano metodę, która pozwala ominąć utrudnienia narzucone programiście przez system operacyjny układu automatyki kompleksowej.

3. Uruchamianie programów specjalnych w warunkach ograniczonej diagnostyki błędów wykonania

Programiści piszący oprogramowanie użytkowe (nie tylko programy specjalne) dla układów automatyki kompleksowej, powinni mieć dostęp w trakcie uruchamiania programów, do wszystkich fortranowych procedur ułatwiających wykrycie i lokalizację błędów. Jednak ze względów, o których była mowa pod koniec poprzedniego rozdziału, nie zawsze tak jest. Testowanie i uruchamianie programów specjalnych trzeba wówczas zorganizować inaczej. Czynności z tym związane można podzielić na dwa etapy:

- testowanie programu specjalnego off line,
- badania eksploatacyjne programu specjalnego, działającego pod kontrolą systemu operacyjnego układu automatyki kompleksowej.

W pierwszym etapie testowany program jest uruchamiany pod kontrolą programu zarządzającego Executive, na standardowym zestawie sprzętu. W ten sposób, programista dysponuje praktycznie całą dostępną pamięcią operacyjną jednostki centralnej, standardowymi urządzeniami zewnętrznymi oraz wszystkimi procedurami pomocniczymi takimi jak TRACE dla Fortranu [1], MONITOR, GTRACE, SPATCH [5] dla Planu.

Programy specjalne korzystają z systemowych podprogramów komunikacji ze zmiennymi procesu i urządzeniami wejścia/wyjścia, dlatego też, podczas kompilacji do programu specjalnego powinny być dołączone procedury symulujące odpowiednie podprogramy systemowe. Zestaw takich procedur symulujących opracowano dla systemu ABAK [4] i zostały one dopisane do grupy SRF7 kompilatora Fortranu XFAM. Instrukcje procedur symulujących są takie same, jak odpowiadające im instrukcje podprogramów systemowych. Różnice polegają przede wszystkim na sposobie korzystania z urządzeń wejścia/wyjścia.

Po zastosowaniu procedur symulujących, wszystkie informacje, które mają być wyprowadzone przez drukarki mozaikowe, są wyprowadzane przez drukarkę wierszową, zaś informacje, które mają być wymieniane przez program specjalny ze zmiennymi procesu, są wczytywane z kart lub taśmy papierowej, ewentualnie wyprowadzane przez drukarkę wierszową lub taśmę papierową. Procedury symulujące powinny być napisane tak, aby tekst programu przeznaczonego do testowania off-line jak najmniej różnił się od tekstu programu docelowego. W tym przypadku, różnica polega na dopisaniu dodatkowych linii sterujących kompilacją, w których deklaruje się urządzenia wejścia-wyjścia.

Przy odpowiednim doborze danych testujących, można w pierwszym etapie uruchamiania programów specjalnych wyeliminować prawie wszystkie lub nawet wszystkie błędy programowe logiczne. Drugi etap uruchamiania polega na sprawdzeniu działania programu specjalnego pod kontrolą systemu operacyjnego. Na tym etapie, trzeba liczyć się z wystąpieniem niezauważonego dotąd błędu. Gdyby przynajmniej w pierwszym okresie eksploatacji programu, należy się przed tym zabezpieczyć.

W przypadku uruchomionego w jednej z lubelskich cukrowni, systemu SORT, takie zabezpieczenie się było utrudnione przez ograniczoną ilość miejsca przeznaczonego w pamięci operacyjnej komputera na programy specjalne i brak kanału wejścia/wyjścia, przez który procedura błędu RACL mogłaby wyprowadzać informacje o błędach. W systemie SORT programy specjalne mogły komunikować się z urządzeniami wejścia/wyjścia tylko poprzez systemowe podprogramy PRINT i READ. Dodatkową niedogodnością były trudności w lokalizacji błędu, który mógł dotyczyć zarówno programu specjalnego, jak i oprogramowania podstawowego.

Program specjalny (napisany w Fortranie) skompilowano więc ponownie z opcją śledzenia TRACE 1 i ze źródłowym podprogramem własnej obsługi błędów, który został dołączony do krajowego systemu wykrywania błędów poprzez podprogram FTRAP [2]. Po wykryciu błędu system wykrywania błędów wywołał podprogram własnej obsługi błędów, przekazując mu numer kodu błędu. Następnie podprogram ten, systemową procedurą PRINT, wyprowadził przez drukarkę mozaikową kod błędu i informacje lokalizujące błąd.

Aby ułatwić odszukanie błędnego fragmentu programu, wprowadzono trzy pomocnicze zmienne: ITRACE, ISEGTR i ISEG. Wartość zmiennej ITRACE, zerowana na początku segmentu głównego, była zwiększana o 1, w wybranych miejscach programu. W szczególności wszystkie instrukcje kończące pętle DO zwiększały wartość zmiennej ITRACE. Na początku każdego segmentu typu FUNCTION lub SUBROUTINE pod zmienną ISEG była podstawiana (znakowo) nazwa tego segmentu, a na końcu wartość zmiennej ISEGTR była zwiększana o jedność. Po wykryciu błędu, wykonania pozostałego błędu, na drukarkę mozaikową były wyprowadzane wartości zmiennych ITRACE, ISEGTR i znakowo ISEG. Ponadto były drukowane wartości wybranych, wytypowanych w programie zmiennych. Znając wartość zmiennej ITRACE można było wskazać fragment programu, w którym wystąpił błąd, a jeśli błąd wystąpił w pętli DO, można było stwierdzić, ile razy pętla była wykonana. Zmienne ISEG i ISEGTR pomagały zlokalizować błąd, który wystąpił w podprogramach, jak również zawężały fragment segmentu głównego, wskazany przez zmienną ITRACE, zawierający błędną instrukcję.

Opisana metoda postępowania zwiększa program o około 300 słów (procedura TRACE 1 + własny podprogram monitorujący błędy), ale wydaje się, że warto ją zastosować nawet kosztem podziału jednego programu specjalnego na dwa mniejsze, aby później łatwiej, a przede wszystkim szybciej usunąć błąd.

Mając dane wejściowe, przy których ten błąd wystąpił, można wrócić do etapu pierwszego i prześledzić sytuację przy pełnym śledzeniu programu.

4. Podsumowanie

W artykule zwrócono uwagę przede wszystkim na diagnostykę błędów w programach specjalnych, ale nie należy zapominać o tym, że również sposób pisania programów ma duży wpływ na liczbę i łatwość lokalizowania popełnionych omyłek.

Każdy program, nie tylko specjalny, powinien być *odporny* na złe dane. Program powinien sygnalizować pojawienie się nieprawidłowych danych, a nie usiłować dzielić przez zero, czy też wyciągać pierwiastek z liczby ujemnej. Mogą również występować dane, które nie spowodują zakończenia się programu w trybie błędu, lecz obliczone na ich podstawie wyniki będą bez sensu. W tym przypadku wydaje się celowe drukowanie wszystkich danych wykorzystywanych przez program.

Programy specjalne, przed ich uruchomieniem pod kontrolą systemu operacyjnego układu automatyki kompleksowej, powinny być sprawdzone off-line na standardowym zestawie komputerowym, z wykorzystaniem procedur symulujących odpowiednie podprogramy systemowe. Pozwala to, na pełniejsze i w dogodniejszych warunkach prowadzone badania programu.

Nawet bardzo dokładnie sprawdzone programy mogą zawierać błędy, które wyjdą na jaw dopiero w trakcie eksploatacji. Dlatego programy specjalne pracujące w układach automatyki kompleksowej powinny mieć możliwość korzystania ze wszystkich lub prawie wszystkich firmowych procedur wykrywania błędów. Możliwości te zależą od oprogramowania podstawowego. Z ich brakiem można sobie poradzić, wprowadzając do programu specjalnego własny podprogram monitujący błędy.

Wydaje się, że spełnienie powyższych postulatów nie tylko ułatwi i przyspieszy uruchamianie programów specjalnych, ale również zwiększy zaufanie użytkownika do układu automatyki kompleksowej jako całości.

Literatura

[1] ICL Fortran Magnetic Compiler 1900 Series System Odra 1300. Elwro, Wrocław.

[2] Fortran — funkcje pomocnicze. Zeszyt 2. Publ. 13043/2, Elwro, Wrocław.

[3] SZPAK, podręcznik programowania. Praca zbiorowa. Sprawozdanie MERA-PIAP nr 1370.

[4] Jurkowski J.: Procedury symulujące bibliotekę podprogramów komunikacji programów specjalnych z systemem operacyjnym ABAK. MERA-PIAP, Warszawa.

[5] Plan — podprogramy. Podprogramy ogólne. Zeszyt 4. Publ. 13030/4, Elwro, Wrocław.