mgr inż. Sławomir GRZECHNIK
Przemysłowy Instytut Automatyki
i Pomiarów MERA–PIAP
Warszawa

ORGANIZACJA PRACY WIELOZADANIOWEJ W WIELODOSTĘPNYM KOMPUTEROWYM SYSTEMIE ZARZĄDZANIA

W artykule zaprezentowano rozwiązanie złożonego problemu programowania z wykorzystaniem pracy wielozadaniowej. Jest to zgodne z obecnymi tendencjami światowymi, na co wskazuje rozwój wielozadaniowych systemów operacyjnych umożliwiających stosowanie przedstawionej metody.

1. WSTĘP

Celem artykułu jest przedstawienie jednej z metod pracy wielozadaniowej zastosowanej do wielodostępnego komputerowego systemu użytkowego. Omawiana metoda została użyta do stworzenia Modułowego Systemu Zarządzania Magazynami Wysokiego Składowania zwanego dalej w skrócie MSZMWS. MSZMWS jest zestawem oprogramowania, w oparciu o które można zbudować duży użytkowy system zarządzania. Stworzone oprogramowanie ma stanowić rdzeń systemu użytkowego i realizować funkcje ściśle związane z systemem operacyjnym komputera, takie jak koordynacja zadań użytkowych, komunikacja między zadaniami, zarządzanie bazą danych, komunikacja z operatorem i inne. System został napisany w języku C na maszynie SM–4 (PDP 11/40) i jest przeznaczony do pracy z systemem operacyjnym RSX11M.

2. CECHY SZCZEGÓLNE DUŻYCH SYSTEMÓW ZARZĄDZANIA

Komputerowe systemy zarządzania znajdują najrozmaitsze zastosowanie w bardzo wielu dziedzinach. Mimo tej różnorodności można jednak wyróżnić w nich pewne cechy wspólne, które odróżniają je od innych systemów oprogramowania. Pierwszą taką podstawową cechą jest praca z dużą bazą danych. Dlatego jednym z zasadniczych zadań systemu jest utrzymywanie informacji w bazie danych w stanie możliwie bliskim rzeczywistości.

Drugą cechą jest wymaganie (na ogół) wielodostępności, co oznacza, że kilkunastu lub więcej użytkowników może korzystać z systemu w tym samym czasie. Każdy z nich oczekuje dość szybkiej reakcji systemu i wymaga dostępu do bazy danych, co pociąga za sobą konieczność koordynacji ich poczynań i takiego oprogramowania, które spełnia żądania użytkownika w jego odczuciu niemal natychmiast. Wielodostępna praca z bazą danych wymaga specjalnych algorytmów dostępu do niej i zmiany informacji w bazie w taki sposób, aby równoległa praca użytkowników nie zakłóciła jej zgodności.
Jako trzecią cechę należy dodać, że od systemów zarządzania oczekuje się spełniania rozlicznych funkcji użytkowych związanych z rejestracją zdarzeń, aktualizacją bazy danych itd. W przeciętnym systemie można wymienić kilkadziesiąt takich funkcji.
Niezwykle ważnym zagadnieniem jest zabezpieczenie bazy danych przed zniszczeniem lub uszkodzeniem spowodowanym dostępem nieuprawnionych osób lub złą pracą sprzętu. Przed tym mają chronić odpowiednie algorytmy dostępu oraz prowadzenie rejestracji zdarzeń umożliwiającej restart.
Należy jeszcze wspomnieć, że system powinien być stosunkowo prosty w użyciu i powinien wspomagać operatora w jego pracy przez narzucenie logiki kolejnych operacji, podpowiadanie, czy wyjaśnienia w postaci programów pomocników („helpów").
To wszystko powoduje, że systemy zarządzania są niezwykle złożonymi systemami oprogramowania, które wymagają specjalnego podejścia od programistów.

3. PODZIAŁ SYSTEMU NA MNIEJSZE ZADANIA

Złożone zagadnienie programowania można rozwiązać dwoma sposobami. Pierwszy z nich polega na stworzeniu jednego, dużego programu, który spełnia wszelkie funkcje wymagane od systemu. Takie rozwiązanie ma bardzo wiele wad, gdyż powstający program jest bardzo skomplikowany, a tym samym trudny do przetestowania i uruchomienia. Części programu bywają ze sobą powiązane w nieoczekiwany sposób, dzięki czemu wprowadzenie zmian w jednej z nich powoduje błędne działanie pozostałych. Komplikacja algorytmu utrudnia nie tylko przetestowanie, ale również ustrzeżenie go przed sytuacjami awaryjnymi (np. błędne dane wejściowe), których liczba rośnie wraz ze stopniem złożoności algorytmu. Ponadto każdorazowe dołączenie nowego podprogramu lub modyfikacja istniejących wymaga konsolidacji (linkowania) całego programu, co dla dużego programu trwa dość długo. Należy jeszcze dodać, że podział programu na nakładki (ang. overlays) też może być trudnym zagadnieniem. Zaletą takiego podejścia jest stosunkowo małe związanie z systemem operacyjnym komputera, konieczne jedynie do bieżącej obsługi terminali systemu. Jest to zaleta dość istotna, gdyż w przypadku napisania programu w języku wyższego rzędu, przeniesienie

go pod inny system operacyjny wymaga na ogół niewielkich zmian. Wydaje się jednak, że to podejście ma więcej wad niż zalet.

Drugim rozwiązaniem jest podział zadania na małe programy, które budowane są niezależnie, ale mają ze sobą współpracować. Małe zadania są prostsze do zaprojektowania i uruchomienia, sama kompilacja i konsolidacja zabiera niewiele czasu zwiększając wydajność programisty. Takie rozwiązanie wymaga dużego nakładu pracy włożonego w zaprojektowanie i uruchomienie mechanizmów współpracy zadań, ich koordynacji i komunikacji. Ponadto sam system oprogramowania zostaje silnie związany z systemem operacyjnym komputera, z którego należy czerpać funkcje równoległego uruchamiania zadań, ich komunikacji itd. Ogranicza to klasę możliwych do wykorzystania systemów operacyjnych do systemów wielozadaniowych wyposażonych w ekstrakody (ang. system calls, system directives) obsługi zdarzeń w czasie rzeczywistym.

Obecnie spotyka się systemy użytkowe budowane w oparciu o jedno i drugie rozwiązanie. W systemach operacyjnych jednozadaniowych stosuje się rozwiązanie pierwsze, w systemach wielozadaniowych – drugie, chociaż bywają wyjątki od tej zasady.

MSZMWS został konsekwentnie zbudowany w oparciu o rozwiązanie drugie, dzięki czemu do jego zalet należy prostota budowy zadań użytkownika przy braku górnego ograniczenia ich liczby, natomiast wadą jest silne związanie z systemem RSX11M. Następny rozdział pracy jest poświęcony krótkiej charakterystyce systemu RSX11M z uwagi na jego fundamentalne znaczenie dla MSZMWS.

4. SYSTEM OPERACYJNY RSX11M

System ten został napisany dla komputerów linii PDP 11 firmy Digital Equipment Corporation. Jest to system wielozadaniowy z tzw. podziałem czasu dla zadań (ang. time sharing) wyposażony w mechanizmy czasu rzeczywistego. Powstawał i był rozwijany w latach 70. gdy pamięć minikomputerów nie przekraczała 256 Kb, co zresztą bardzo dodatnio wpłynęło na koncepcję systemu. System składa się ze stosunkowo niedużego jądra obudowanego warstwami zadań systemowych, dobieranych zgodnie z wymaganiami użytkownika w procesie zwanym generacją.

Liczba zadań użytkownika, które mogą pracować równolegle, jest większa niż wynika to z dostępnej pamięci operacyjnej dzięki zastosowaniu mechanizmu chwilowego wyładowania na dysk zadań tracących dostęp do CPU (procesora). Algorytm przydziału CPU uwzględnia priorytety zadań i wiele innych okoliczności, co powoduje jego dużą złożoność. Odpowiednia generacja zapewnia, że przydział CPU odbywa się według zaleceń użytkownika.

Rezydentna część systemu zajmuje stosunkowo niewiele pamięci operacyjnej. System wymaga więcej pamięci tylko na czas wykonywania żądanych zadań systemowych. Około 8 Kb pamięci zajmowanej przez system operacyjny stanowi tzw. pamięć dynamiczną systemu, gdzie przechowywane są tablice zadań aktywnych, pakiety we/wy itd. Wraz z podprogramami obsługi urządzeń zewnętrznych i rezydentną częścią podsystemu zbiorów system zajmuje około 60 Kb pamięci. Reszta pamięci jest dynamicznie wykorzystywana przez zadania użytkowe i niekiedy przez nierezydentne zadania systemowe.

Niezwykle ważną właściwością systemu jest wykorzystywanie techniki pamięci wirtualnej. Polega to na tym, że adresowanie wewnątrz każdego zadania jest ograniczone do 64 Kb. Ten zakres jest nazywany pamięcią wirtualną zadania i zadanie nie może go przekroczyć. Po załadowaniu zadania do pamięci jego pamięć wirtualna zostaje odwzorowana na konkretne adresy fizycznego obszaru przydzielonego dla zadania. Odbywa się to za pomocą specjalnego układu sprzętowego w trybie pracy „kernel" procesora. Dzięki temu zadanie, mające przydzielony obszar pamięci fizycznej, nie może go przekroczyć i np. zniszczyć innych zadań, czy samego systemu operacyjnego, leżących w pamięci komputera. Zabezpiecza to zarówno przed skutkami błędów oprogramowania jak i złymi intencjami programistów.

Istnieje też mechanizm pozwalający odwzorować część pamięci wirtualnej zadania na dynamicznie utworzone obszary pamięci fizycznej zwane regionami. Do takiego regionu kilka zadań może przypisać swą pamięć wirtualną uzyskując w ten sposób obszar wspólny do przekazywania dużych ilości danych do innych zadań.

W systemie RSX11M istnieją bogate mechanizmy do rozpoznawania i obsługi zdarzeń. Służą do tego celu semafory (ang. flags) i asynchroniczne pułapki systemowe zwane w skrócie AST (ang. asynchronuous system traps). Semafory mogą być włączane i wyłączane przez poszczególne zadania w celu sygnalizacji zdarzeń, ale ważniejsza jest możliwość powiązania ich ze zdarzeniami systemowymi; wtedy zapalane są one przez system operacyjny, gdy dane zdarzenie wystąpiło. Zdarzeniami systemowymi są np. zakończenia wszelkich operacji we/wy, zakończenie pracy innego zadania, wysyłanie komunikatów, upływ okresu czasu i wiele innych. Zadania zainteresowane pewnymi zdarzeniami mogą sprawdzać stan odpowiednich semaforów lub zawiesić się do czasu zapalenia semafora. W tym drugim przypadku zadanie jest odwieszane przez system operacyjny. Dzięki temu zadania mogą synchronizować swą pracę zdarzeniami zewnętrznymi. Pułapki AST dają możliwość natychmiastowej reakcji na zdarzenie systemowe. Zadanie, które chce obsłużyć jakieś zdarzenie (np. zakończenie pracy innego zadania), specyfikuje obsługę AST związanej z tym zdarzeniem (jest to podprogram napisany w odpowiedni sposób i dołączony do zadania). Gdy zdarzenie zajdzie, to praca zadania zostanie natychmiast przerwana i wykonana zostanie obsługa AST, po czym wykonywanie zadania zostanie podjęte w punkcie przerwania.

Semafory i AST umożliwiają obsługę zdarzeń w kolejności ich wystąpienia, jak również według priorytetów ustalonych przez programistę.
System umożliwia tworzenie bibliotek rezydentnych zawierających powszechnie używane podprogramy. Kod tych podprogramów rezyduje w wydzielonym obszarze pamięci, a zadania z nich korzystające odwzorowują na ten obszar część swojej pamięci wirtualnej. Te podprogramy nie są dołączane do zadań na etapie konsolidacji, dzięki czemu same zadania są odpowiednio mniejsze.
Należy również wspomnieć o systemie zbiorów zwanym FILES–11. Zbiory są umieszczane na urządzeniach zewnętrznych w katalogach (ang. directories), które nie tworzą drzewiastej struktury jak w nowszych systemach (np. system UNIX, gdzie taka struktura została zastosowana po raz pierwszy). Zestaw operacji na zbiorach jest bardzo bogaty i zawiera operacje dotyczące bloków fizycznych urządzeń oraz rekordów logicznych. Do systemu operacyjnego nie są dołączone operacje wyższego rzędu jak dostęp indeksowo–sekwencyjny, sortowanie, itd.

5. DEKOMPOZYCJA MODUŁOWEGO SYSTEMU ZARZĄDZANIA

Jak wspomniano poprzednio, MSZMWS powstał w oparciu o rozwiązanie polegające na podzieleniu złożonego zagadnienia na mniejsze, powiązane ze sobą funkcjonalnie. W sumie na system składa się około 40. zadań stanowiących jądro przyszłego systemu użytkowego. Te zadania są podzielone na kilka grup, które można nazwać podsystemami. Oto one wraz z krótką charakterystyką:

a. Podsystem zarządzania terminalami i zadaniami.
   Stanowią go dwa zadania FOREST i MAIN. Pierwsze z nich wywoływane jest na początku dziennej pracy systemu i inicjalizuje zbiory i tablice systemowe. Zadanie MAIN natomiast rezyduje stale w pamięci, prowadzi nasłuch terminali i uruchamia na bieżąco zadania użytkownika zgłaszane z terminali. Zadania użytkownika będą dalej nazywane dyrektywami operatorskimi.
b. Podsystem zarządzania bazą danych.
   Koordynuje dostęp zadań użytkowych do zbiorów bazy danych i prowadzi w czasie rzeczywistym rejestrację istotnych zdarzeń, a w szczególności historię wszelkich operacji na bazie danych umożliwiającą ewentualny restart. Zadanie główne tego podsystemu nazywa się DBM. Zadanie pomocnicze, uruchamiane w razie potrzeby, nazywa się UNFOLD i służy do wycofywania błędnych operacji na bazie danych.
c. Uprzywilejowane dyrektywy operatorskie.
   Są to pojedyńcze programy zwane dyrektywami, gdyż uruchamiane są po wprowadzeniu przez operatora ich nazwy z terminala. Stanowią język komend systemu.

Określenie uprzywilejowane oznacza, że służą one do zmiany stanu systemu, np. do włączania i odłączania terminali, nadawania i zmiany uprawnień terminali, oglądania istotnej informacji o systemie, itd. Mogą być uruchamiane tylko z uprzywilejowanych terminali systemu.

d. Podsystem restartu.
Pracuje w trybie off–line i w wypadku awarii służy do odtworzenia stanu bazy danych na podstawie jej kopii i historii wszystkich operacji na bazie danych wykonanych od czasu utworzenia tej kopii. Historia ta jest prowadzona przez zadanie DBM.

e. Zadania pomocnicze.
W wersji obecnej jest to zadanie HELP mogące wspomagać operatora na bieżąco i zadania służące do rozbudowy zbiorów HELP. W tej grupie mogą pojawić się inne zadania na życzenie użytkownika.

f. Zadania użytkownika.
Zadania użytkownika będą dyrektywami operatorskimi, czyli zostaną włączone do języka komend systemu. W przeciętnym systemie zarządzania jest ich kilkadziesiąt. W MSZMWS nie ma górnego ograniczenia ich liczby. Zadania użytkownika pisane są jak zwykle programy, z tym, że należy używać w nich pewnych specjalnych podprogramów sprzęgających je z systemem. Wszystkie muszą wywoływać podprogram o nazwie start (), który między innymi zwiąże ich standardowe wejście i wyjście z właściwym terminalem., przetworzy ewentualne parametry z wiersza wywołania i stworzy okno adresowe w ich pamięci wirtualnej do komunikacji z zadaniem DBM. Ponadto istnieją jeszcze trzy podprogramy do współpracy z zadaniem DBM.

Dyrektywy operatorskie mogą pracować w trzech środowiskach:

- Praca normalna pod kontrolą zadania MAIN.
Parametry wejściowe są pobierane z wiersza wywołania dyrektywy i ewentualnej konwersacji z operatorem. Komunikacja z zadaniem DBM w celu dostępu do bazy danych jest obowiązkowa.
- Praca pod kontrolą zadania restartu.
Parametry wejściowe, które w czasie odtwarzanej sesji były wprowadzone z terminala, teraz są przekazywane przez zadanie restartu na podstawie zbiorów przechowujących ślad operacji tworzonych w czasie sesji. Nie ma komunikacji z DBM.
- Praca pod kontrolą systemu operacyjnego (poza MSZMWS).
Ta możliwość jest wykorzystywana w czasie uruchamiania i testowania dyrektyw bez potrzeby uruchamiania MSZMWS. Może służyć również do przeglądania zbiorów bazy danych czy innych prac pomocniczych poza pracą MSZMWS.

Możliwość pracy w trzech środowiskach jest zapewniona przez podprogram start ().

Rozpoznaje on środowisko wywołania dyrektywy, odpowiednio przywiązuje standardowe wejście i wyjście i ustawia semafory, z których korzystają podprogramy współpracy z DBM. Te podprogramy również dostosowują swą pracę do środowiska wywołania. Dzięki odpowiedniemu zaprojektowaniu tych podprogramów programista piszący dyrektywę nie musi jej sam dopasowywać do różnych środowisk i pisze ją w sposób standardowy jak do pracy pod MSZMWS.

6. ZASOBY I ZBIORY SYSTEMOWE MSZMWS

Zasobami podstawowymi są: pamięć komputera, urządzenia dyskowe i terminale. Mogą być dołączone i inne urządzenia jak drukarki i taśmy magnetyczne, ale nie są one niezbędne do pracy systemu.

W pamięci komputera na początku pracy jest tworzony i wypełniany region dynamiczny o nazwie FOREST (około 4 Kb), który przechowuje następujące dane reprezentujące chwilowy stan systemu:

- tablica terminali zawierająca informację o terminalach aktualnie dołączonych do systemu, jak numery fizyczne urządzeń, przydzielone stanowisko logiczne określające uprawnienia, semafor do sygnalizacji zdarzeń, wskaźnik stanu zablokowania i odblokowania;
- tablica dyrektyw zawierająca kody dostępności dyrektyw ze stanowisk logicznych i wskaźnik pracy równoległej (zezwalający na jednoczesną pracę terminali). Ponadto dla dyrektyw aktualnie aktywnych odnotowany jest numer przydzielonego terminala i nazwy pod jakimi są chwilowo zainstalowane w systemie operacyjnym;
- dane globalne, jak czas rozpoczęcia sesji, aktualne hasło, identyfikatory urządzeń, na których prowadzone są zbiory systemowe, itd.

Region FOREST jest dostępny wyłącznie dla głównych zadań systemowych jak MAIN, DBM i UNFOLD i dla uprzywilejowanych dyrektyw operatorskich.

W czasie normalnej sesji używane są następujące zbiory obrazujące przebieg sesji:

- TRACE.DAT – zawierający historię operacji na bazie danych. Zbiór ten stanowi podstawę do odtworzenia przebiegu sesji przez zadanie restartu, gdy zajdzie taka potrzeba.
- DZD.DAT – do przechowywania pełnej informacji o aktualnie wykonywanej operacju na bazie danych. Umożliwia odtworzenie stanu bazy danych sprzed operacji, gdy w trakcie jej wykonywania wystąpiły błędy.
- ERRFILE.DAT – chronologiczna informacja o wszelkich błędach, które wystąpiły w czasie sesji.

- CONFIG.DAT — zadanie zamykające sesję; zapisuje w nim informację o aktualnej konfiguracji systemu. Przy następnym uruchomieniu systemu konfiguracja może być odtworzona z tego zbioru.

7. PRACA ZADAŃ W RÓŻNYCH FAZACH SESJI

W początkowej fazie sesji są przygotowywane zasoby potrzebne do pracy MSZMWS. Przywiązanie urządzeń logicznych do fizycznych odbywa się w sposób interakcyjny za pomocą języka komend MCR (Monitor Console Routines) systemu operacyjnego. Pozostałe czynności przygotowawcze, jak otwarcie zbiorów, ustalenie hasła, inicjalizacja terminali, itd. wykonywane są przez zadanie FOREST, które po zakończeniu pracy jest usuwane z pamięci operacyjnej. Faza wstępna kończy się po przejęciu kontroli przez zadanie MAIN i uruchomieniu zadania DBM.
W czasie normalnej pracy sesji zadanie MAIN jest zadaniem nadrzędnym (ang. parent task) wszystkich innych zadań, które są przez nie kreowane i zwane zadaniami potomnymi (ang. off–spring tasks). Praca zadania MAIN polega na zarządzaniu terminalami i kreowaniu zadań–dyrektyw zgłaszanych przez operatorów. Obsługa zdarzeń z tym związanych odbywa się w czasie rzeczywistym. Kreacja każdej dyrektywy poprzedzona jest szeregiem testów sprawdzających możliwość jej uruchomienia. Na czas swojej pracy dyrektywa otrzymuje nadzór nad terminalem. Zadanie MAIN reaguje na zakończenie pracy dyrektywy przejmując jej terminal i usuwając ją z pamięci komputera.
Wszystkie dyrektywy mające wykonywać operacje na bazie danych komunikują się z zadaniem DBM, które koordynuje dostęp do bazy i zabezpiecza operacje w sposób umożliwiający odtworzenie stanu bazy w przypadku błędów. DBM odbiera zlecenia od dyrektyw w czasie rzeczywistym i tworzy kolejkę zadań czekających na dostęp do bazy.
Zakończenie pracy sesji odbywa się na żądanie operatora na stanowisku głównym systemu. Dyrektywy czekające na dostęp do bazy danych są usuwane i tylko dyrektywa aktualnie pracująca na bazie może zakończyć swe działanie normalnie. Gdy to nastąpi, informacja o stanie systemu jest zapisywana do odpowiednich zbiorów i sesja zostaje zamknięta. Ponadto wykonywana jest kopia bazy danych.
Awaryjne zakończenie pracy jest wykonywane automatycznie w przypadku uszkodzeń sprzętu lub niespodziewanej przerwy w pracy zadania DBM. W tej sytuacji należy na podstawie kopii bazy danych z poprzedniej sesji i historii operacji na bazie z bieżącej sesji do czasu awarii wykonać restart doprowadzający bazę do stanu aktualnego.

8. UWAGI KOŃCOWE

Najpoważniejszą wadą przedstawionej metody jest brak przenośności oprogramowania ze względu na uzależnienie od systemu operacyjnego. Problem ten prawdopodobnie będzie w przyszłości rozwiązany poprzez standaryzację mechanizmów pracy wielozadaniowej i włączenie ich do języka programowania, który byłby powszechnie akceptowany. Przykładem takiego języka jest język ADA. Inną możliwością jest powstanie uniwersalnego systemu operacyjnego zawierającego wspomniane mechanizmy, dzięki czemu sprawa wyboru języka programowania byłaby nieistotna.

Literatura

[1] RSX11M v. 4.1. Executive Reference Manual Digital Equipment Corporation. Maynard, Massachussetts.

[2] RSX11M Operator's Procedures Manual. Digital Equipment Corporation, Maynard, Massachussetts.

[3] RSX11M System Generation and Management Guide. Digital Equipment Corporation, Maynard, Massachussetts.

[4] RSX11M I/O Operations Reference Manual. Digital Equipment Corporation, Maynard, Massachussetts.

*
* *