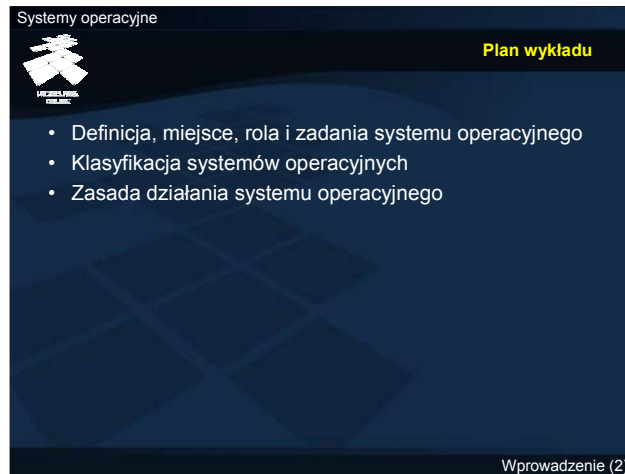
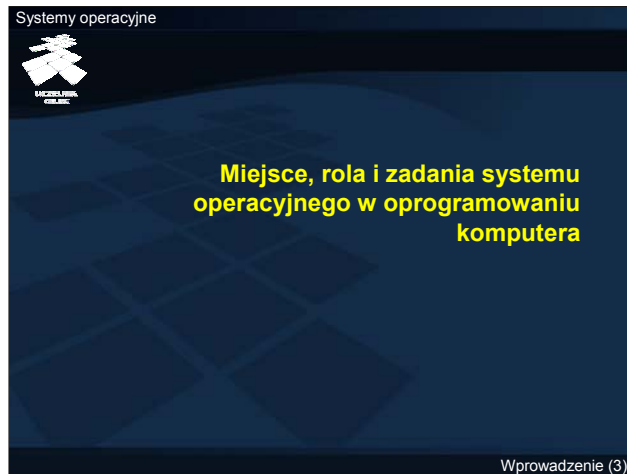


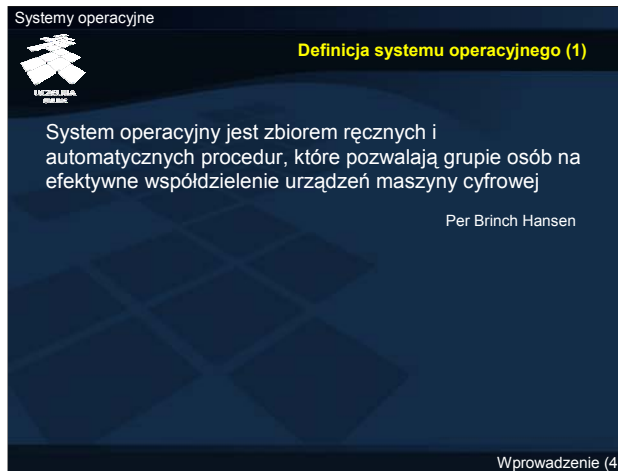


Celem wykładu jest przedstawienie ogólnych informacji o systemie operacyjnym jako składowej oprogramowania komputera. Omawiana jest zatem jego rola i zadania, klasyfikacja oraz specyficzny sposób wykonywania, wynikający ze ścisłej integracji z poziomem maszynowym procesora.



Wykład rozpoczyna się od przedstawienia kilku ogólnych definicji systemu operacyjnego, z których wynika jego rola i miejsce w oprogramowaniu komputera. Omówiona jest też ogólna struktura systemu operacyjnego. Następnie przedstawiona jest klasyfikacja systemów operacyjnych według różnych kryteriów. Ostatnia część dotyczy wykonywania systemu operacyjnego jako programu w komputerze, czyli przekazywania sterowania do kodu jądra. Poruszone przy tej okazji są kwestie zabezpieczenia jądra systemu przed niekontrolowaną ingerencją ze strony oprogramowania aplikacyjnego.





Pojęcie systemu operacyjnego jest trudne do zdefiniowania w zwartej, lakonicznej formie. Najczęściej krótki opis jest zbyt ogólny, żeby uzyskać wyobrażenie o roli i sposobie działania tego specyficznego oprogramowania. Podobnie, trudne jest określenie elementów składowych systemu operacyjnego, czyli jednoznaczne oddzielenie oprogramowania systemowego od aplikacyjnego.

Na kolejnych slajdach przedstawione są bardzo ogólne definicje systemu operacyjnego, podane przez autorów najważniejszej monografii z tej dziedziny. Warto podkreślić, że definicje te pojawiały się na przestrzeni około 40 lat, co odzwierciedla w pewnym sensie sposób postrzegania roli i zadań systemu operacyjnego.

Systemy operacyjne



Definicja systemu operacyjnego (2)

System operacyjny (nadzorczy, nadrzędny, sterujący) jest to zorganizowany zespół programów, które pośredniczą między sprzętem a użytkownikami, dostarczając użytkownikom zestawu środków ułatwiających projektowanie, kodowanie, uruchamianie i eksploatację programów oraz w tym samym czasie sterują przydziałem zasobów dla zapewnienia efektywnego działania.

Alan Shaw

Wprowadzenie (5)

Systemy operacyjne

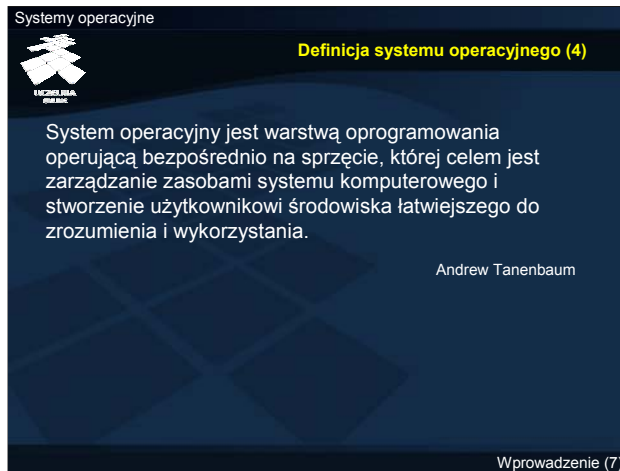


Definicja systemu operacyjnego (3)

System operacyjny jest programem, który działa jako pośrednik między użytkownikiem komputera a sprzętem komputerowym. Zadaniem systemu operacyjnego jest tworzenie środowiska, w którym użytkownik może wykonywać programy w sposób wygodny i wydajny.

Abraham Silberschatz

Wprowadzenie (6)



Systemy operacyjne

Definicja systemu operacyjnego (4)

System operacyjny jest warstwą oprogramowania operującą bezpośrednio na sprzęcie, której celem jest zarządzanie zasobami systemu komputerowego i stworzenie użytkownikowi środowiska łatwiejszego do zrozumienia i wykorzystania.

Andrew Tanenbaum

Wprowadzenie (7)

W niemal wszystkich tych definicjach przewijają się dwie zasadnicze kwestie:

- zarządzanie zasobami systemu komputerowego,
- stworzenie środowiska, wygodnego dla użytkownika.



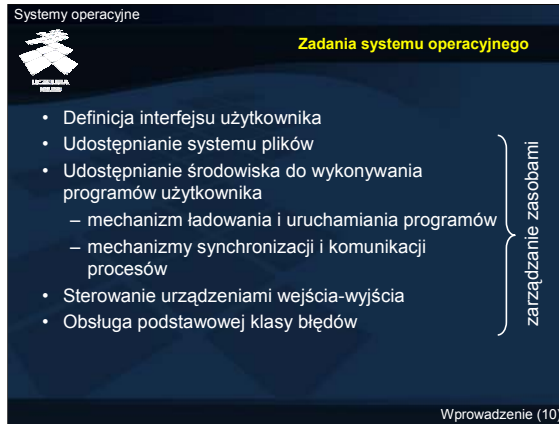
System operacyjny pośredniczy pomiędzy użytkownikiem a sprzętem, dostarczając wygodnego środowiska do wykonywania programów. Użytkownik końcowy korzysta z programów (aplikacji), na potrzeby których przydzielane są zasoby systemu komputerowego. Przydziałem tym zarządza system operacyjny, dzięki czemu można uzyskać stosunkowo duży stopień niezależności programów od konkretnego sprzętu oraz odpowiedni poziom bezpieczeństwa i sprawności działania.



Nie ma precyzyjnego określenia, które składniki wchodzą w skład systemu operacyjnego jako jego części. Najczęściej akceptuje się definicję „marketingową”, zgodnie z którą to wszystko, co producent udostępnia w ramach zbioru oprogramowania nazywanego *systemem operacyjnym*, stanowi jego część. (Podejście takie było przyczyną wielu problemów prawnych firmy Microsoft).

W ogólnym przypadku w strukturze systemu operacyjnego wyróżnia się jądro oraz programy systemowe, które dostarczane są razem z systemem operacyjnym, ale nie stanowią integralnej części jądra. Jądro jest zbiorem modułów, które ukrywają szczegóły sprzętowej realizacji systemu komputerowego, udostępniając pewien zestaw usług, wykorzystywanych między innymi do implementacji programów systemowych. W dalszej części system operacyjny będzie rozumiany głównie jako jądro, ewentualnie inne elementy oprogramowania integralnie związane z funkcjonowaniem jądra.

Z punktu widzenia kontaktu z użytkownikiem istotny jest interpreter poleceń, który może być częścią jądra lub programem systemowym (np. w systemie UNIX). Interpreter wykonuje pewne polecenia wewnętrznie, tzn. moduł lub program interpretera dostarcza implementacji tych poleceń. Jeśli interpreter nie może wykonać wewnętrznie jakiegoś polecenia, uruchamia odpowiedni program (tzw. polecenie zewnętrzne), jako odrębny proces.



Systemy operacyjne

Zadania systemu operacyjnego

- Definicja interfejsu użytkownika
- Udostępnianie systemu plików
- Udostępnianie środowiska do wykonywania programów użytkownika
 - mechanizm ładowania i uruchamiania programów
 - mechanizmy synchronizacji i komunikacji procesów
- Sterowanie urządzeniami wejścia-wyjścia
- Obsługa podstawowej klasy błędów

zarządzanie zasobami

Wprowadzenie (10)

Definiowanie interfejsu użytkownika: system operacyjny dostarcza użytkownikom zbiór poleceń lub system okienkowy wraz z odpowiednim menu, który umożliwia interakcję z systemem komputerowym.

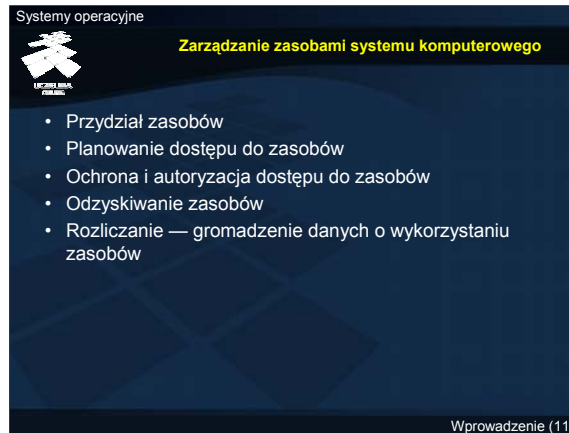
Udostępnianie systemu plików: system operacyjny organizuje i ułatwia dostęp do informacji np. w postaci hierarchicznego systemu plików.

Udostępnianie środowisko do wykonywania programów: system operacyjny dostarcza struktur danych do organizacji wykonywania programu oraz zachowywania i odtwarzania stanu przetwarzania (procesy i przełączanie kontekstu). System operacyjny udostępnia też programistom mechanizmy komunikacji pomiędzy procesami (kolejki komunikatów, strumienie, pamięć współdzielona) oraz mechanizmy synchronizacji procesów (np. semafony).

Sterowanie urządzeniami wejścia-wyjścia: odpowiednie moduły sterujące, integrowane z systemem operacyjnym, inicjalizują pracę urządzeń zewnętrznych oraz pośredniczą w efektywnym przekazywaniu danych pomiędzy jednostką centralną a tymi urządzeniami.

Obsługa podstawowej klasy błędów: system operacyjny reaguje na błędy użytkowników (np. niedostępność zasobów, brak prawa dostępu), programistów (np. błąd dzielenia przez 0, naruszenie ochrony pamięci, nieprawidłowy kod rozkazu) oraz systemu (np. błąd braku strony, błąd magistrali).

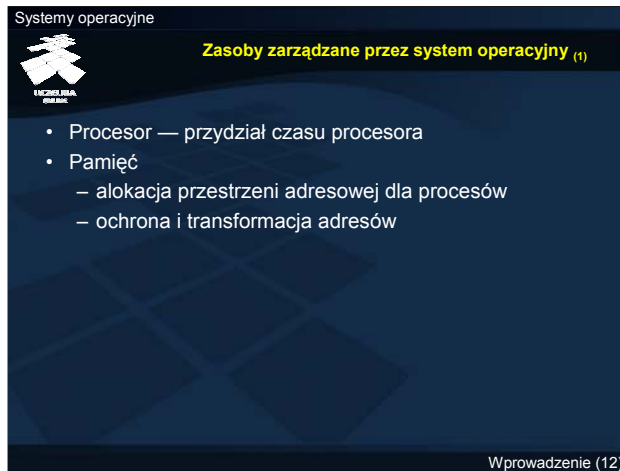
Efektywność zarządzania zasobami oraz wygodny interfejs dla użytkownika są dwoma ogólnymi, niezależnymi celami projektowymi systemów operacyjnych. Pierwszy z tych celów był kluczowy w rozwoju rodziny systemów uniksowych. Dopiero w późniejszych etapach ich rozwoju pojawił się intuicyjny okienkowy interfejs użytkownika. Systemy rodziny MS Windows zorientowane były natomiast przede wszystkim na interfejs użytkownika, na bazie którego w późniejszych etapach rozwoju powstawał pełnowartościowy system operacyjny, uwzględniający szerzej rozumiane zarządzanie zasobami.



Formalne pojęcie zasobu zostanie wprowadzone wraz z pojęciem procesu. Na razie zasób będzie rozumiany intuicyjnie jako element systemu komputerowego istotny, czy wręcz kluczowy dla realizacji przetwarzania. Funkcja zarządzania zasobami nie jest bezpośrednio wykorzystywana przez użytkownika (czasami nie jest przez niego w ogóle dostrzegana). Jej celem jest optymalizacja wykorzystania zasobów przez użytkowników.

W ramach zarządzania ogólnie rozumianymi zasobami można wyróżnić następujące operacje:

- Przydział zasobów: realizacja żądań dostępu do zasobów w taki sposób, że zasoby używane są zgodnie z intencją użytkowników (np. zagwarantowanie wyłącznego dostępu drukarki).
- Planowanie dostępu do zasobów: strategia przydziału zasobów gwarantująca bezpieczeństwo, żywotność, brak zakleszczenia, sprawiedliwość oraz optymalność ich wykorzystania. Warto zwrócić uwagę na odróżnienie planowania od samego przydziału — przydział oznacza powiązanie zasobu z realizowanym zadaniem, podczas gdy planowanie wiąże się z podejmowaniem decyzji odnośnie wyboru zdania, któremu zasób będzie przydzielony.
- Ochrona i autoryzacja dostępu do zasobów: dopuszczanie możliwości użytkowania zasobu tylko przez osoby uprawnione i w zakresie przydzielonych im uprawnień.
- Odzyskiwanie zasobów: dołączanie zwolnionych zasobów do zbioru zasobów wolnych po zakończeniu ich użytkowania.
- Rozliczanie: rejestrowanie i udostępnianie informacji o wykorzystaniu zasobów w celach kontrolnych i rozrachunkowych.

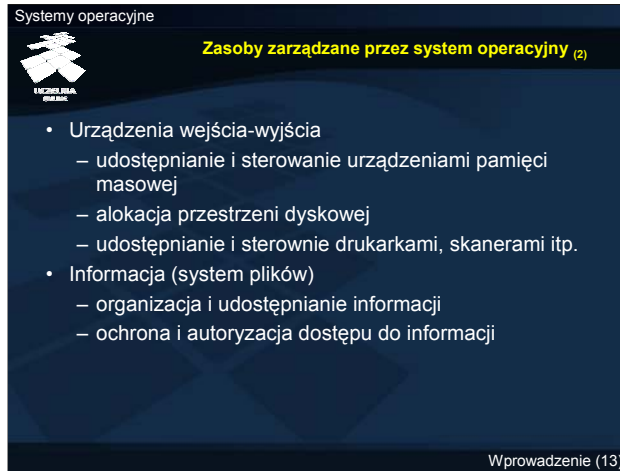


Typowymi zasobami zarządzanymi przez system operacyjny są: procesor, pamięć, urządzenia wejścia-wyjścia (w tym system plików, stanowiący tzw. wirtualne wejście-wyjście). Zależnie od zasobu zarządzanie charakteryzuje się pewną specyfiką.

Procesor jest zasobem współdzielonym przez wiele procesów, w związku z czym zadaniem systemu operacyjnego jest przydział kwantu czasu procesora i wywłaszczanie zadania, które:

- wykorzystało już swój czas lub
- nie może kontynuować przetwarzania ze względu na brak innych zasobów (np. brak gotowości urządzeń wejścia-wyjścia) lub też
- ma zbyt niski priorytet.

Pamięć jest zasobem, który przydzielany jest na wyłączność danego przetwarzania. Zadaniem systemu jest zatem utrzymywanie informacji o zajętości przestrzeni adresowej, znajdowanie i przydzielanie odpowiednich fragmentów wolnej pamięci na żądanie aplikacji użytkownika lub innych modułów systemu operacyjnego oraz reagowanie na naruszenie ochrony pamięci. System operacyjny pośredniczy również w transformacji adresów wirtualnych na fizyczne w systemach z segmentacją lub stronicowaniem przez organizację tablicy segmentów lub stron oraz obsługę błędów strony.



Systemy operacyjne

Zasoby zarządzane przez system operacyjny (2)

- Urządzenia wejścia-wyjścia
 - udostępnianie i sterowanie urządzeniami pamięci masowej
 - alokacja przestrzeni dyskowej
 - udostępnianie i sterowanie drukarkami, skanerami itp.
- Informacja (system plików)
 - organizacja i udostępnianie informacji
 - ochrona i autoryzacja dostępu do informacji

Wprowadzenie (13)

Urządzenia zewnętrzne są stosunkowo wolne, w związku z czym, w celu poprawy efektywności, zarządzanie tymi urządzeniami wymaga odpowiedniej organizacji systemu przerwań oraz buforowania danych, ewentualnie spoolingu. Osobnym zagadnieniem jest zarządzanie urządzeniami pamięci masowej, zwłaszcza odpowiednia organizacja przestrzeni dyskowej oraz optymalizacja ruchu głowic dyskowych.

Informacja jest z punktu widzenia użytkownika najważniejszym zasobem, gdyż jej przetwarzanie jest celem systemu komputerowego. System operacyjny odpowiada za organizację gromadzenia i udostępniania informacji, jej ochronę przed nieuprawnioną ingerencją, spójność w przypadku awarii itp.



Systemy operacyjne



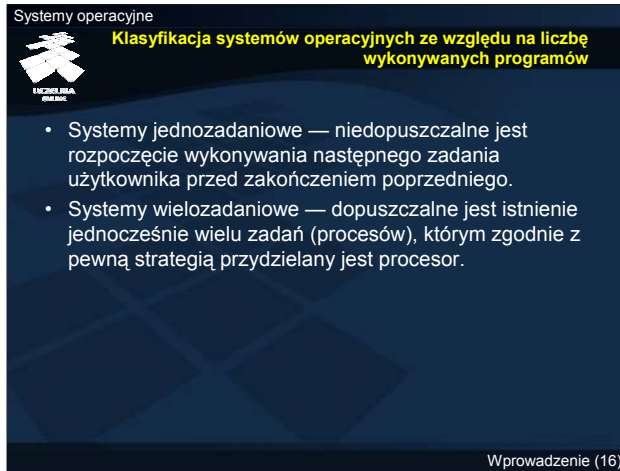
Klasyfikacja systemów operacyjnych ze względu na sposób przetwarzania

- Systemy przetwarzania bezpośredniego (ang. on-line processing systems) — systemy interakcyjne
 - występuje bezpośrednia interakcja pomiędzy użytkownikiem a systemem,
 - wykonywanie zadania użytkownika rozpoczyna się zaraz po przedłożeniu.
- Systemy przetwarzania pośredniego (ang. off-line processing systems) — systemy wsadowe
 - występuje znacząca zwłoka czasowa między przedłożeniem a rozpoczęciem wykonywania zadania,
 - niemożliwa jest ingerencja użytkownika w wykonywanie zadania.

Wprowadzenie (15)

W przypadku systemu przetwarzania bezpośredniego użytkownik wprowadza zadanie do systemu i oczekuje na wyniki. W trakcie przetwarzania jest zatem możliwa interakcja pomiędzy użytkownikiem a systemem (aplikacją). Użytkownik może być na przykład poproszony o wprowadzenie jakiś danych na terminalu, wybranie czegoś z menu itp.

W przypadku przetwarzania pośredniego zadanie jest realizowane w czasie wybranym przez system. Po przedłożeniu zadania ingerencja użytkownika jest niemożliwa. Wszystkie dane muszą być zatem dostępne w momencie przedkładania zadania, a jakikolwiek błąd programowy (np. niekompletność danych) oznacza konieczność przedłożenia i wykonania zadania ponownie.



Systemy operacyjne

Klasyfikacja systemów operacyjnych ze względu na liczbę wykonywanych programów

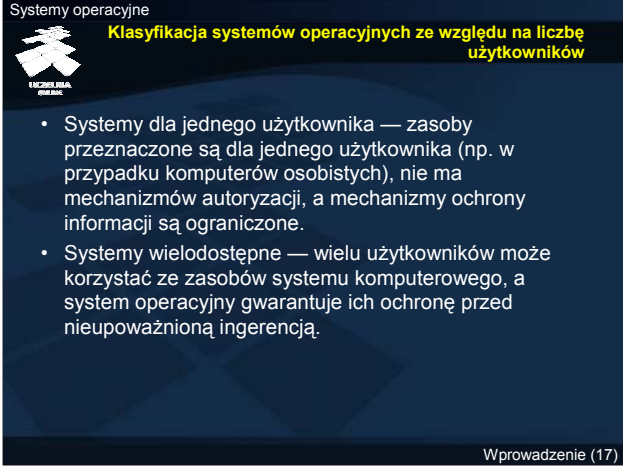
- Systemy jednozadaniowe — niedopuszczalne jest rozpoczęcie wykonywania następnego zadania użytkownika przed zakończeniem poprzedniego.
- Systemy wielozadaniowe — dopuszczalne jest istnienie jednocześnie wielu zadań (procesów), którym zgodnie z pewną strategią przydzielany jest procesor.

Wprowadzenie (16)

Systemy jednoprogramowe, zwane też jednozadaniowymi, umożliwiają uruchomienie jednego zadania użytkownika, które ewentualnie może być wykonywane współbieżnie z pewnymi procedurami systemu operacyjnego.

Systemy wieloprogramowe (wielozadaniowe) dostarczają mechanizm przełączania kontekstu, umożliwiając w ten sposób zachowanie stanu wykonywania określonego programu (stanu procesu), a następnie odtworzenie stanu wykonywania innego programu (w szczególności innego wykonywania tego samego programu). Przełączenie kontekstu jest skutkiem zwolnienia procesora, które z kolei następuje w wyniku:

- żądania przydziału dodatkowego zasobu,
- zainicjowania operacji wejścia-wyjścia,
- przekroczenia ustalonego limitu czasu (kwantu czasu),
- uzyskania gotowości przez inne zadanie (proces) o wyższym priorytecie.



Systemy operacyjne

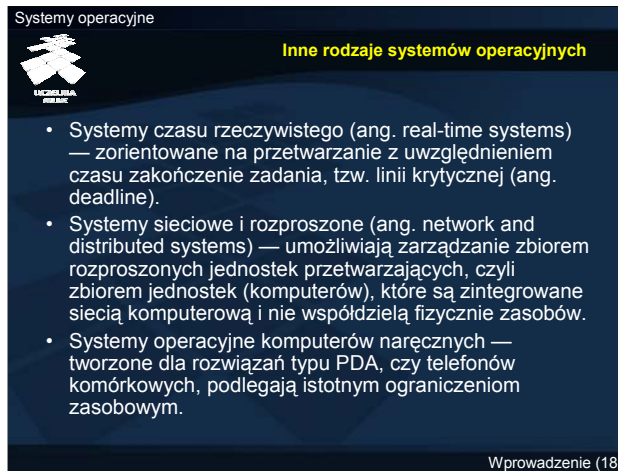
Klasyfikacja systemów operacyjnych ze względu na liczbę użytkowników

- Systemy dla jednego użytkownika — zasoby przeznaczone są dla jednego użytkownika (np. w przypadku komputerów osobistych), nie ma mechanizmów autoryzacji, a mechanizmy ochrony informacji są ograniczone.
- Systemy wielodostępne — wielu użytkowników może korzystać ze zasobów systemu komputerowego, a system operacyjny gwarantuje ich ochronę przed nieupoważnioną ingerencją.

Wprowadzenie (17)

W systemach dla jednego użytkownika nie ma problemu autoryzacji, czyli konieczności identyfikowania zleceniodawcy poszczególnych zadań. Mechanizmy ochrony są ograniczone w tym sensie, że nie ma potrzeby ochrony zasobów jednego użytkownika przed drugim użytkownikiem tego samego systemu operacyjnego, ale w czasie powszechności sieci rozległych istnieje jednak problem ochrony zasobów przed ingerencją z zewnątrz.

System operacyjny w przypadku wielodostępu musi zagwarantować, że jeden użytkownik nie jest w stanie zakłócić pracy innych użytkowników. Jest to problem właściwego udostępniania zasobów oraz dostępności mechanizmów ochrony „prywatnych” zasobów jednego użytkownika przed ingerencją innego.



Systemy operacyjne

Inne rodzaje systemów operacyjnych

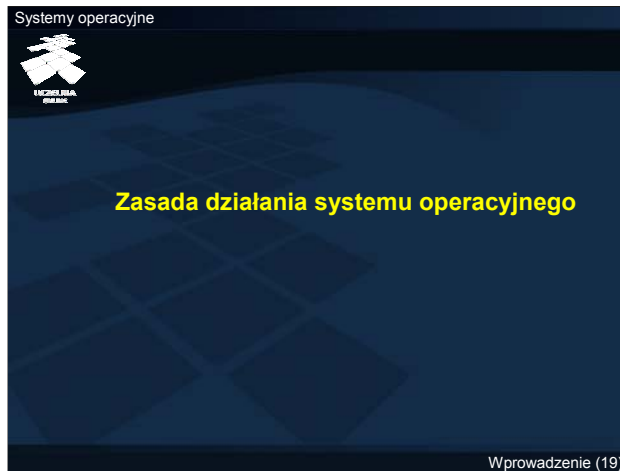
- Systemy czasu rzeczywistego (ang. real-time systems) — zorientowane na przetwarzanie z uwzględnieniem czasu zakończenia zadania, tzw. linii krytycznej (ang. deadline).
- Systemy sieciowe i rozproszone (ang. network and distributed systems) — umożliwiają zarządzanie zbiorem rozproszonych jednostek przetwarzających, czyli zbiorem jednostek (komputerów), które są zintegrowane siecią komputerową i nie współdzielą fizycznie zasobów.
- Systemy operacyjne komputerów naręcznych — tworzone dla rozwiązań typu PDA, czy telefonów komórkowych, podlegają istotnym ograniczeniom zasobowym.

Wprowadzenie (18)

W systemach czasu rzeczywistego priorytetem jest minimalizacja czasu odpowiedzi (reakcji) lub czasu realizacji zadania, gdyż po przekroczeniu pewnego czasu wartość wyników albo jest znacznie mniejsza (np. przewidywanie kursów akcji na giełdzie) albo są one całkowicie bezużyteczne (np. prognozowanie pogody). Szczególnym przypadkiem, gdzie czas jest krytyczny, są wszelkiego rodzaju systemy sterowania w czasie rzeczywistym (np. w komputerach pokładowych samochodów, samolotów, jednostek pływających itp.). Systemy operacyjne czasu rzeczywistego są więc budowane pod kątem szybkości reakcji na zdarzenie zewnętrzne. Ich zadaniem jest minimalizować czas oczekiwania na zasoby dla czasowo krytycznych zadań, dlatego unika się w ich przypadku rozwiązań, które zmniejszają przewidywalność tego czasu (np. pamięci wirtualnej).

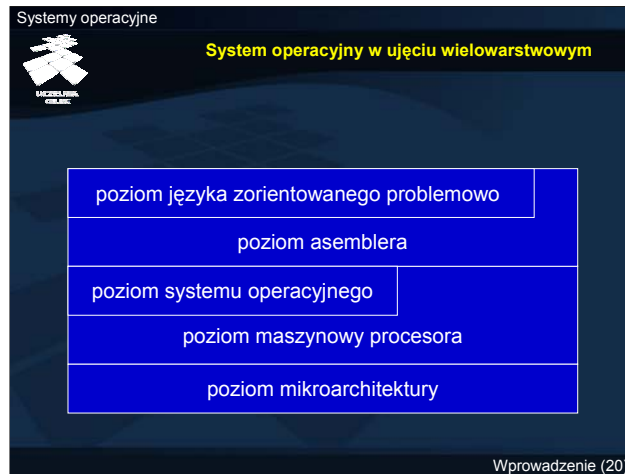
Rozproszone systemy operacyjne zapewniają, że system komputerowy, złożony z autonomicznych jednostek przetwarzających połączonych siecią komputerową, postrzegany jest jako całość. Zasoby tego systemu udostępniane są w jednolity sposób niezależnie od ich fizycznej lokalizacji — niezależnie od tego, czy są to zasoby lokalne danej jednostki, czy zasoby związane integralnie z jednostką zdalną. Cecha ta odróżnia systemy rozproszone od systemów sieciowych, które również umożliwiają dostęp do zdalnych zasobów, ale nie ukrywają faktu fizycznego rozproszenia tych zasobów. Inaczej mówiąc, w systemie sieciowym odróżnia się dostęp lokalny i dostęp zdalny do zasobów.

Rozwiązania dla systemów naręcznych nie muszą tworzyć środowiska dla zaawansowanego przetwarzania wielozadaniowego, ale ze względu na niewielkie rozmiary urządzeń podlegają dość rygorystycznym ograniczeniom zasobowym. W przypadku tego typu rozwiązań, jak również rozwiązań dla innych urządzeń mobilnych, dość istotnym zasobem, którym należy odpowiednio zarządzać jest energia.



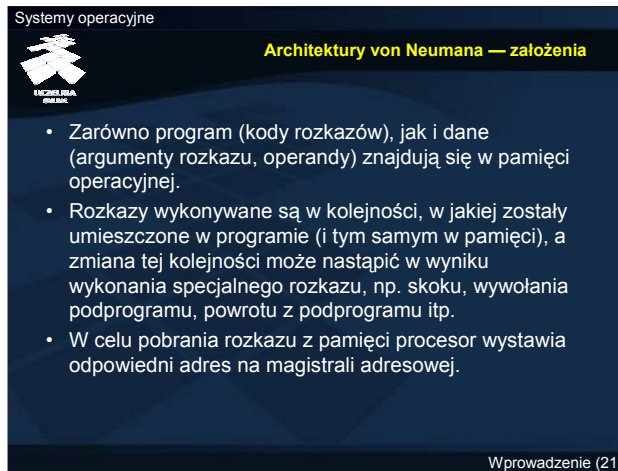
System operacyjny jest programem, jednak jego działanie jest dość specyficzne, gdyż musi on nadzorować (monitorować) pracę komputera nawet wówczas, gdy wykonywany jest jakiś program aplikacyjny. Systemy operacyjne muszą reagować na błędy w programach aplikacyjnych, porządkować system komputerowy po awariach, z kolei błędy w kodzie jądra systemu operacyjnego mogą zdestabilizować funkcjonowanie całego systemu komputerowego.

Działanie współczesnych systemów operacyjnych jest rezultatem ewolucji w architekturze sprzętowo-programowej, w której potrzeby w zakresie implementacji pewnych mechanizmów systemu operacyjnego wymuszały wprowadzanie stosownych rozwiązań na poziomie architektury komputera (procesora, jednostki zarządzania pamięcią, układu bezpośredniego dostępu do pamięci, procesorów wejścia-wyjścia itp.). Rozwiązania na poziomie architektury komputera otwierały z kolei drogę do dalszego rozwoju oprogramowania systemowego.



Działanie systemu komputerowego można opisywać na różnych poziomach abstrakcji, począwszy od zjawisk fizycznych na poziomie układów półprzewodnikowych, czy też propagacji sygnałów logicznych na poziomie układów techniki cyfrowej. Tak niski poziom abstrakcji jest jednak na ogół mało interesujący dla informatyka, dlatego na najniższym poziomie abstrakcji na slajdzie umieszczona została mikroarchitektura. Poziom mikroarchitektury jest jednak zastrzeżony dla twórców procesorów, natomiast dla programistów systemów komputerowych najniżej dostępny jest poziom maszynowy procesora. Na poziomie tym definiowana jest lista rozkazów procesora, tryby adresowania pamięci, rejestry procesora. Na poziomie tym nie istnieją jednak takie elementy, jak pliki, procesy, mechanizmy komunikacji i synchronizacji. Te elementy uzupełniane są przez system operacyjny, który współtworzy wraz poziomem maszynowym hybrydową warstwę usług dla programów użytkowych. Na bazie tej warstwy budowane są kolejne poziomy abstrakcji, związane z językami programowania niższego lub wyższego poziomu.

W praktyce rzadko kiedy korzysta się bezpośrednio z poziomu maszynowego. Jeśli rzeczywiście istnieje potrzeba pisania programu na tak niskim poziomie, wykorzystywany jest raczej assembler, który nie ogranicza możliwości poziomu maszynowego, a usprawnia tworzenie programu dzięki mnemonikom rozkazów zamiast ich kodów, etykietom zamiast adresów itp.



Systemy operacyjne

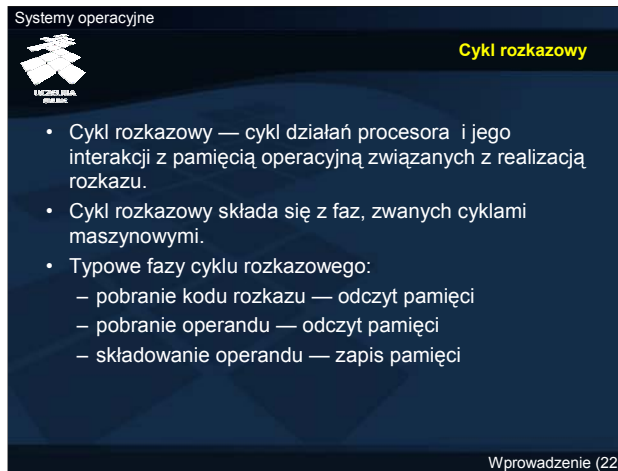
Architektury von Neumana — założenia

- Zarówno program (kody rozkazów), jak i dane (argumenty rozkazu, operandy) znajdują się w pamięci operacyjnej.
- Rozkazy wykonywane są w kolejności, w jakiej zostały umieszczone w programie (i tym samym w pamięci), a zmiana tej kolejności może nastąpić w wyniku wykonania specjalnego rozkazu, np. skoku, wywołania podprogramu, powrotu z podprogramu itp.
- W celu pobrania rozkazu z pamięci procesor wystawia odpowiedni adres na magistrali adresowej.

Wprowadzenie (21)

W celu wyjaśnienia, w jaki sposób wykonywany jest program jądra, istotne jest uświadomienie sobie, w jaki sposób w ogóle wykonywany jest program przez procesor.

Działanie współczesnych procesorów opiera się w dużej części na modelu von Neumana. Architektura komputera, której nazwa przyjęła się od nazwiska jej popularyzatora — Johna von Neumana, zakłada, że zarówno dane, jak i program (kod instrukcji/rozkazów) znajdują się w pamięci operacyjnej (dziś wydaje się to dość oczywiste). Rozkazy umieszczane są pod kolejnymi adresami w pamięci. Wykonywanie takiego programu sprowadza się zatem do pobierania rozkazów z kolejnych komórek. Adres komórki pamięci, od której rozpoczyna się kod następnego rozkazu do wykonania, przechowywany jest w odpowiednim rejestrze procesora, zwanym licznikiem programu (PC — program counter) lub wskaźnikiem instrukcji (IP — instruction pointer). Zawartość tego rejestru wystawiana jest na szynę adresową magistrali systemowej w celu pobrania z pamięci kodu rozkazu. Po zdekodowaniu operacji licznik ten zwiększany jest odpowiednio do długości pobranego rozkazu, w ten sposób wskazuje następny rozkaz do wykonania. Opisany schemat — domyślny przepływ sterowania — oznacza wykonywanie rozkazów w pewnej sekwencji, wynikającej z ich uporządkowania w programie i tym samym w pamięci. Schemat ten może ulec zmianie w wyniku wykonania specjalnego rozkazu (skoku, wywołania podprogramu, powrotu z podprogramu). Zmiana domyślnego przepływu sterowania jest więc zdefiniowana przez sam program.



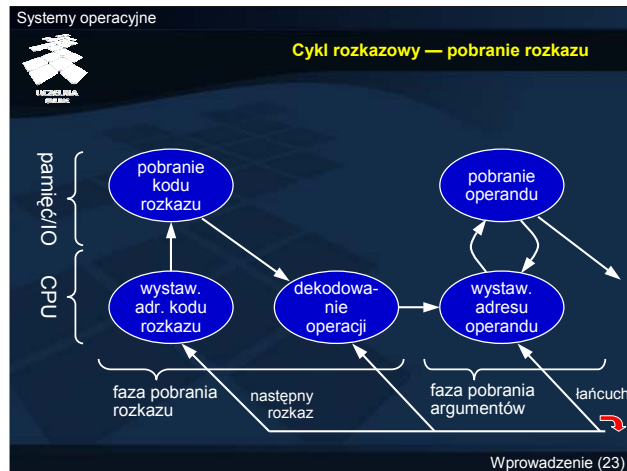
Systemy operacyjne

Cykl rozkazowy

- Cykl rozkazowy — cykl działań procesora i jego interakcji z pamięcią operacyjną związanych z realizacją rozkazu.
- Cykl rozkazowy składa się z faz, zwanych cyklami maszynowymi.
- Typowe fazy cyklu rozkazowego:
 - pobranie kodu rozkazu — odczyt pamięci
 - pobranie operandu — odczyt pamięci
 - składowanie operandu — zapis pamięci

Wprowadzenie (22)

Działania procesora, zmierzające do wykonania rozkazu powtarzają się cyklicznie, w związku z czym określa się je jako cykl rozkazowy. Realizacja cyklu rozkazowego wymaga na ogół kilku interakcji procesora z pamięcią. Każdą taką interakcję określa się mianem cyklu maszynowego. W każdym cyklu rozkazowym występuje cykl maszynowy pobrania kodu rozkazu (fetch). W zależności od trybu dostępności operandów mogą też wystąpić cykle pobrania operandu z pamięci (albo rejestrów wejścia-wyjścia) lub składowania operandu w pamięci (albo rejestrach wejścia-wyjścia). (Operandy są argumentami operacji wykonywanej w ramach rozkazu.) Każdy cykl maszynowy oznacza zatem zapis lub odczyt pamięci, przy czym cykl pobrania kodu rozkazu oznacza zawsze odczyt.



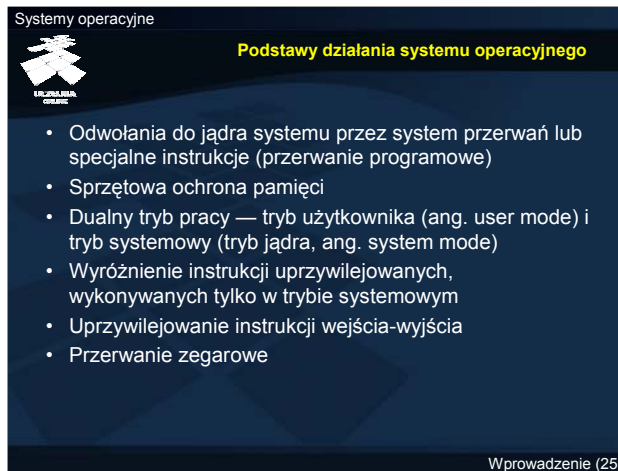
Cykl rozkazowy rozpoczyna się od wystawienia kodu rozkazu. W reakcji na towarzyszący temu sygnał sterujący odczytu pamięci na szynie danych udostępniana jest zawartość zaadresowanej komórki, a procesor pobiera ją i zapamiętuje w odpowiednim rejestrze. Następnie zostaje zdekodowana operacja i w zależności od wymaganych operandów następuje odczyt pamięci lub rejestrów wejścia-wyjścia. Odczyt taki wymaga oczywiście wystawienia odpowiedniego adresu na szynie adresowej, a następnie przekazania sygnału sterującego odczytu pamięci. Adres operandu w zależności od trybu adresowania jest częścią kodu rozkazu lub znajduje się w rejestrze procesora (rzadziej w innej komórce pamięci). W przypadku rozkazów wymagających kilku operandów wejściowych cykl maszynowy pobrania operandów może powtórzyć się kilkakrotnie. W szczególnym przypadku rozkaz nie wymaga żadnego operandu wejściowego. Może się też okazać, że operand dostępny jest w rejestrze procesora, w związku z czym nie jest wymagana interakcja z pamięcią albo rejestrami wejścia-wyjścia. W takich przypadkach cykl maszynowy pobrania operandów zostanie pominięty.



Po zdekodowaniu operacji i skompletowaniu operandów wejściowych można wykonać operację, a następnie umieścić w pamięci albo rejestrach wejścia-wyjścia operandy wyjściowe. Podobnie jak w przypadku operandów wejściowych ten cykl maszynowy może być wykonany wielokrotnie lub w szczególnym przypadku pominięty. Na tym kończy się ciąg działań związany z wykonaniem bieżącego rozkazu i można by przejść do następnego cyklu rozkazowego.

Jeśli w taki sposób byłby wykonywany program użytkownika, to nasuwa się pytanie: „Gdzie jest miejsce na wykonywanie programu jądra systemu operacyjnego?”, „W jaki sposób następuje przekazanie sterowania do jądra systemu operacyjnego?”.

W czasie wykonywania rozkazu mogły nastąpić pewne zdarzenia zewnętrzne w stosunku do procesora, nie związane z bieżącym cyklem rozkazowym, ale wymagające od procesora jakiejś reakcji. Konieczność reakcji zgłaszana jest poprzez sygnał na odpowiedniej linii wejściowej procesora. Ostatnia faza cyklu rozkazowego polega zatem na sprawdzeniu, czy wystąpiło takie zgłoszenie. Jeśli nie było zgłoszenia, rozpoczyna się następny cykl maszynowy. Jeśli jednak było zgłoszenie — nazywane *przerwaniem*, następuje ciąg działań, zmierzających do zidentyfikowania źródła przerwania, a następnie przekazania sterowania do stosownej procedury obsługi. Procedury obsługi przerwania są częścią programu jądra systemu operacyjnego.



Systemy operacyjne

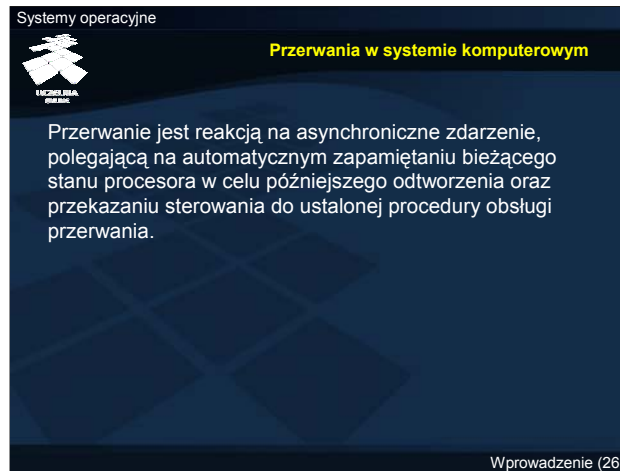
Podstawy działania systemu operacyjnego

- Odwołania do jądra systemu przez system przerwań lub specjalne instrukcje (przerwanie programowe)
- Sprzętowa ochrona pamięci
- Dualny tryb pracy — tryb użytkownika (ang. user mode) i tryb systemowy (tryb jądra, ang. system mode)
- Wyróżnienie instrukcji uprzywilejowanych, wykonywanych tylko w trybie systemowym
- Uprzywilejowanie instrukcji wejścia-wyjścia
- Przerwanie zegarowe

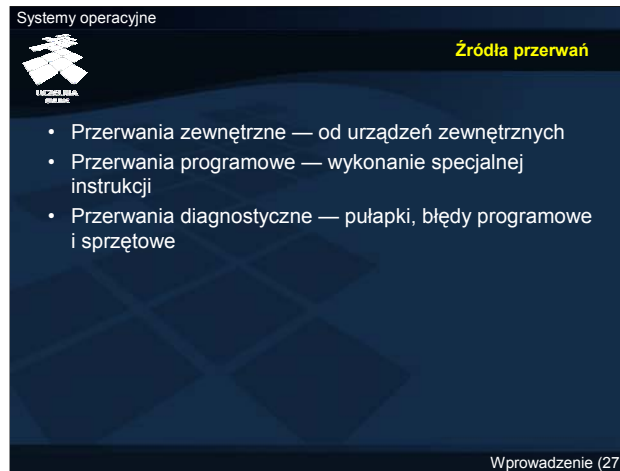
Wprowadzenie (25)

Ogólnie, sterowanie przekazywane jest do jądra systemu operacyjnego poprzez przerwania. Program jądra jest więc zbiorem procedur obsługi przerwań i wywoływanych przez nie innych podprogramów. Przerwania, wspomniane na poprzednim slajdzie, pochodzą z układów na zewnątrz procesora, czyli od urządzeń wejścia-wyjścia, czasomierzy, układu bezpośredniego dostępu do pamięci itp. Inny rodzaj to przerwania zgłaszane wewnętrznie przez procesor, będące następstwem wykrycia jakiegoś stanu wyjątkowego. Jeszcze inny rodzaj to przerwania programowe, wynikające z wykonania specjalnej instrukcji procesora, umożliwiające programom użytkownika dostęp do wybranych funkcji jądra systemu operacyjnego.

Stabilność pracy systemu wymaga ochrony przynajmniej jądra systemu operacyjnego przed niekontrolowaną ingerencją użytkowników. Wymaga to monitorowania odniesień do pamięci i weryfikowania poprawności adresów. Ze względów wydajnościowych zadanie to realizowane jest sprzętowo, ale odpowiednie dane na potrzeby weryfikacji musi dostarczyć system. W celu zabezpieczenia tych (i innych) niewrażliwych danych wyróżnione są pewne instrukcje uprzywilejowane, niedostępne dla programów aplikacyjnych. Powstaje jednak problem odróżnienia programów systemowych od aplikacyjnych, którego rozwiązaniem jest wyodrębnienie dwóch (w niektórych procesorach nawet większej liczby) poziomów pracy (trybów pracy). Możliwe staje się narzucenie sprzętowych restrykcji odnośnie wykonywania niektórych instrukcji na odpowiednich poziomach. Proces użytkownika uruchamiany jest w trybie nieuprzywilejowanym, w związku z czym nie może wykonać pewnych instrukcji, dostępnych tylko w trybie uprzywilejowanym, tym samym ma ograniczoną możliwość swobodnego ingerowania w „obszary” zastrzeżone dla jądra systemu operacyjnego.



System przerwania umożliwia niesekwencyjne (współbieżne) wykonywanie programów. Zmiana sekwencji wykonywania instrukcji polega na tym, że w reakcji na przerwanie następuje zapamiętanie bieżącego stanu przetwarzania (najważniejszych rejestrów procesora), przekazanie sterowania do ustalonej procedury obsługi i rozpoczęcie wykonywania instrukcji tej procedury. W szczególności może to prowadzić do przełączenia kontekstu, czyli przekazania sterowania po zakończeniu procedury obsługi przerwania do innego przetwarzania, niż to które zostało przerwane.

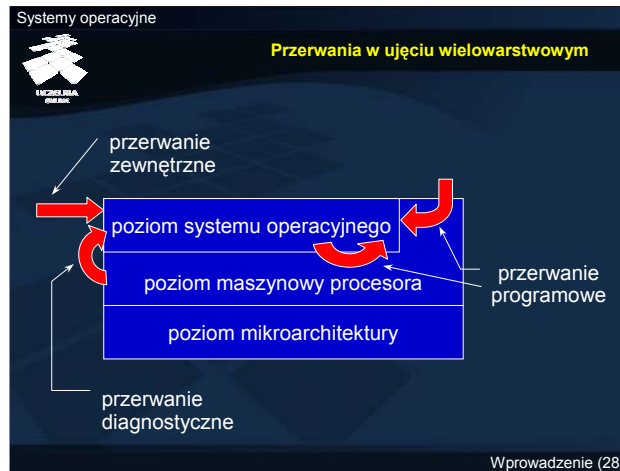


Przerwania od urządzeń zewnętrznych zgłaszane są po zakończeniu operacji wejścia-wyjścia i przekazywane na specjalne wejście procesora najczęściej przez sterownik przerw. Tą samą ścieżką zgłaszane są również przerwania od układów ściśle współpracujących z procesorem — czasomierzy, układów bezpośredniego dostępu do pamięci itp. Są to typowe przerwania, gdyż ich źródło jest poza procesorem i jest od niego niezależne.

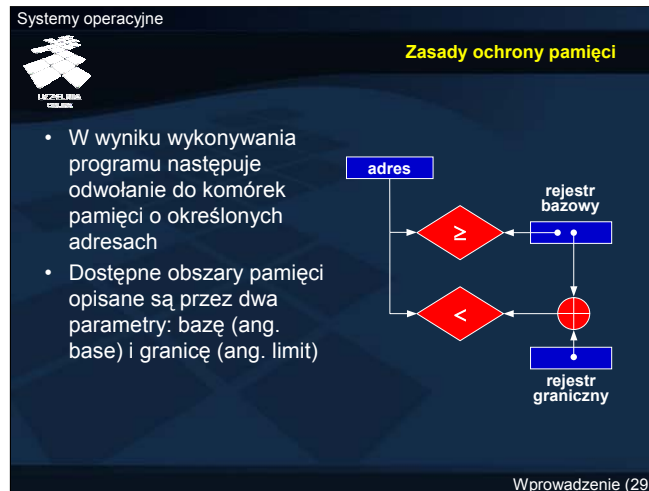
W przeciwieństwie do przerw zewnętrznych, przerwania programowe są wynikiem wykonania specjalnej instrukcji procesora, np. `int` (interrupt) w procesorach firmy Intel, `sc` (system call) w procesorach PowerPC firm IBM, Motorola i Apple.

Przerwania diagnostyczne są z kolei generowane wewnętrznie przez procesor w sytuacji zajścia określonego stanu. Są zatem pośrednim skutkiem wykonania określonego ciągu rozkazów prowadzących do osiągnięcia tego stanu. Tego typu przerwania w literaturze określa się jako pułapki lub wyjątki. Przykładami tego typu przerw są:

- pułapki które pojawiają się, gdy licznik rozkazów osiągnie określoną wartość, tzn. będzie wskazywał na instrukcję, na której założono pułapkę. Mechanizm ten wykorzystywany jest w konstrukcji debugger'ów.
- błędy programowe, typu błąd dzielenia przez 0, naruszenie ochrony pamięci, nieprawidłowy format rozkazu procesora. W reakcji na to przerwanie jądro najczęściej usuwa proces, który je spowodował.
- błędy sprzętowe, wymagające odpowiedniej obsługi ze strony systemu operacyjnego, np. błąd braku strony w przypadku systemów z pamięcią stronicowaną. Zadaniem jądra jest doprowadzenie systemu do takiego stanu, żeby wznowienie tego samego rozkazu nie spowodowało ponownie błędu.



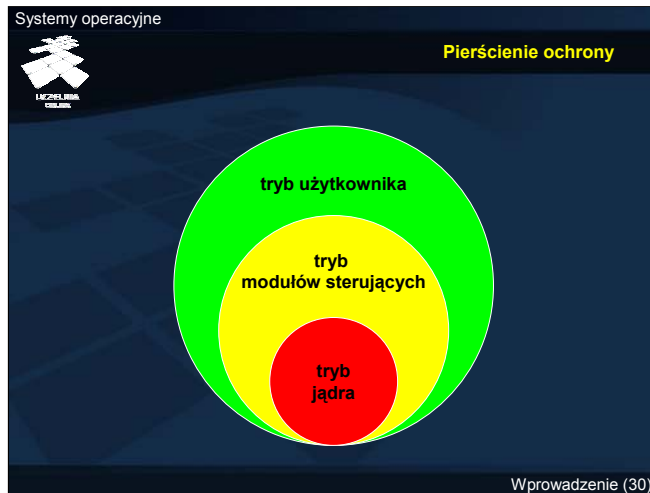
Przerwanie diagnostyczne ma swoje źródło na poziomie maszynowym procesora. Przerwanie programowe też ma swoje źródło na poziomie maszynowym procesora, ale bezpośrednią przyczyną jego wystąpienia jest rozkaz w programie wykonywanym przez procesor. Rozkaz taki najczęściej jest w programie aplikacyjnym, ale może również być w kodzie jądra systemu operacyjnego. Przerwanie zewnętrzne zgłaszane jest procesorowi poprzez podanie odpowiedniego sygnału na specjalne wejście. Procesor może mieć kilka takich wejść — w najprostszym przypadku ma jedno wejście przerwania maskowalnych (przerwanie zgłaszane na tym wejściu można ignorować) oraz jedno wejście przerwania niemaskowalnych (nie można go zignorować).



Kluczowa dla stabilnej pracy i bezpieczeństwa systemu komputerowego jest ochrona pamięci jądra systemu operacyjnego przed ingerencją programów wykonywanych w trybie użytkownika — przede wszystkim przed modyfikacją kodu lub danych, ale ze względów bezpieczeństwa również przed odczytem (np. haseł). W systemach wielozadaniowych ważna jest również ochrona pamięci jednego zadania (procesu) przed ingerencją innego zadania.

W najprostszym przypadku ochrona pamięci polega na ograniczeniu zakresu dostępnych adresów do pewnego podzbioru, opisanego przez podanie najniższego i najwyższego dopuszczalnego adresu. W przedstawionym schemacie najniższy dopuszczalny adres przechowywany jest w rejestrze bazowym. Rejestr graniczny z kolei określa wielkość dostępnego obszaru pamięci. Dopuszczalne adresy należą do zatem do przedziału $< \text{baza}, \text{baza} + \text{granica}$). Jeśli żądany adres jest spoza tego przedziału następuje zgłoszenie przerwania diagnostycznego i sterowanie przekazywane jest do odpowiedniej procedury systemu operacyjnego. W wyniku wykonania tej procedury następuje najczęściej zakończenie wykonywania programu, który spowodował naruszenie ochrony pamięci.

W środowisku wielozadaniowym z każdym przetwarzaniem mogą być związane inne ograniczenia na dostępność obszarów pamięci. Przełączanie między zadaniami wymaga zatem zmiany zawartości rejestrów, ograniczających zakres dostępności pamięci. W rozważaniach pominięte zostały też kwestie ograniczonego dostępu do określonych obszarów (np. tylko do odczytu). Dostępność obszaru w takim przypadku zależy od rodzaju cyklu maszynowego. Tego typu zagadnienia będą rozważane przy omawianiu zarządzania pamięcią.



Dla procesora program jest ciągiem instrukcji. Na tym poziomie nie ma rozróżnienia użytkownika, administratora, operatora itd. Narzucenie restrykcji odnośnie wykonywania niektórych instrukcji wymaga zatem wskazania trybu pracy (związanego z poziomem uprzywilejowania), który musi być czytelny dla procesora w celu weryfikowania dostępności instrukcji.

W najprostszym przypadku wystarczą dwa tryby, ale większa ich liczba może usprawnić tworzenie oprogramowania systemowego. Na rysunku wyszczególniono 3 tryby, zwane też pierścieniami ochrony. W architekturze Intel IA-32 wyróżniono 4 pierścienie. W trybie najbardziej uprzywilejowanym (trybie jądra, poziomie nr 0 w procesorach Intel) dostępne są wszystkie instrukcje i rejestry procesora. W każdym następnym (mniej uprzywilejowanym) trybie jest coraz mniej dostępnych instrukcji lub rejestrów.

Procesor pracuje zatem zawsze w jednym z trybów uprzywilejowania. Programu jądra wykonywany jest w trybie najbardziej uprzywilejowany, z którego można się przełączyć w tryb mniej uprzywilejowany, co ma miejsce przy uruchamianiu kod aplikacji użytkownika. Powrót do trybu uprzywilejowanego możliwy jest poprzez odpowiednie przerwania (lub podobne mechanizmy), ale procedura obsługi przerwania dostarczona jest przez jądro. Program użytkownika nie może zatem zmienić trybu pracy na potrzeby wykonania dowolnego własnego kodu.

Systemy operacyjne

Przerwanie zegarowe

- Przerwanie zegarowe generowane jest przez czasomierz (ang. timer) po wyznaczonym okresie czasu.
- Obsługa przerwania zegarowego oznacza przekazanie sterowania do jądra systemu operacyjnego, umożliwiając w ten sposób wykonanie pewnych zadań okresowych.

Wprowadzenie (31)

Czasomierz jest licznikiem, na wejście którego podawane są impulsy o stałej częstotliwości, niezależnej od częstotliwości zegara procesora. Do licznika wpisywana jest pewna wartość, która zmniejszana jest o 1 z każdym impulsem. Po osiągnięciu wartości 0 generowane jest przerwanie, zwane zegarowym. Jest to jedno z przerw zewnętrżnych. Wystąpienie tego przerwania nazywany jest *taktem zegara* i pojawia się zależnie od architektury kilkadziesiąt lub kilkaset razy na sekundę (zwykle około 100 razy).

W wyniku przerwania zegarowego system operacyjny przejmuje sterowanie, wykonuje pewne zadania okresowe (np. zbiera dane statystyczne w celu optymalizacji pracy systemu) i podejmuje decyzje, czy wznowić przerwane zadanie, czy przełączyć się na inne zadanie (dokonać przełączenia kontekstu). W ten sposób niemożliwe jest zawłaszczenie procesora przez program użytkownika.