Systemy Operacyjne

Ogólna struktura

Arkadiusz Chrobot

Katedra Systemów Informatycznych, Politechnika Świętokrzyska w Kielcach

Kielce, 17 października 2024

Plan wykładu

1. Jeszcze jedna definicja systemu operacyjnego
2. Elementy (podsystemy) systemu operacyjnego
3. Usługi systemu operacyjnego
4. Wywołania systemowe
5. Programy systemowe
6. Struktury jądra sytemu operacyjnego
7. Projektowanie i implementacja systemu operacyjnego

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

Plan wykładu

1. Jeszcze jedna definicja systemu operacyjnego
2. Elementy (podsystemy) systemu operacyjnego
3. Usługi systemu operacyjnego
4. Wywołania systemowe
5. Programy systemowe
6. Struktury jądra sytemu operacyjnego
7. Projektowanie i implementacja systemu operacyjnego

Plan wykładu

1. Jeszcze jedna definicja systemu operacyjnego
2. Elementy (podsystemy) systemu operacyjnego
3. Usługi systemu operacyjnego
4. Wywołania systemowe
5. Programy systemowe
6. Struktury jądra sytemu operacyjnego
7. Projektowanie i implementacja systemu operacyjnego

Plan wykładu

1. Jeszcze jedna definicja systemu operacyjnego
2. Elementy (podsystemy) systemu operacyjnego
3. Usługi systemu operacyjnego
4. Wywołania systemowe
5. Programy systemowe
6. Struktury jądra sytemu operacyjnego
7. Projektowanie i implementacja systemu operacyjnego

System operacyjny — inne spojrzenie

Podobnie jak nie ma jednoznacznej definicji *czym* jest system operacyjny, tak nie ma jednoznacznej definicji *co* nim jest. Termin *system operacyjny* może oznaczać „to co dostarcza producent jako system operacyjny” i obejmować swoim znaczeniem zbiór takich elementów oprogramowania jak: jądro systemu, interpreter poleceń, edytory tekstu itd. Może również określać część oprogramowania systemowego stale rezydującą w pamięci operacyjnej komputera i wykonywaną w trybie monitora procesora, czyli *jądro systemu operacyjnego*. W trakcie tego wykładu system operacyjny będziemy definiować zgodnie z tą drugą możliwością. Wszelkie odstępstwa od tej definicji będą sygnalizowane.

Elementy systemu operacyjnego

Choć istniejące systemy operacyjne różnią się od siebie, to można wyróżnić pewne wspólne elementy, które prawie każdy z nich zawiera. Zaliczają się do nich:

1. podsystem zarządzania procesami,
2. podsystem zarządzania pamięcią operacyjną,
3. podsystem zarządzania pamięcią pomocniczą,
4. podsystem wejścia-wyjścia,
5. system plików,
6. podsystem obsługi sieci,
7. *ochrona*.

Procesy

Każdy uruchomiony i wykonywany program komputerowy jest procesem, który w szczególności może tworzyć inne procesy. Każda praca w systemie komputerowym jest wykonywana w ramach określonego procesu. Aby procesy mogły wykonywać swoje zadania muszą dysponować określonymi zasobami. Te zasoby udostępnia im system operacyjny. Do jego zadań należy również ochrona zasobów przed nieprawidłowym użyciem ich przez procesy. Pojedynczy proces jest wykonywany sekwencyjnie, natomiast kilka procesów może być wykonywanych współbieżnie. Koordynacja takiego wykonania jest również zadaniem systemu operacyjnego.

Obsługa procesów

Czynności, które system operacyjny wykonuje zarządzając procesami obejmują:

- tworzenie i usuwanie procesów,
- wstrzymywanie i wznawianie wykonania procesów,
- zapewnianie możliwości synchronizacji procesów,
- zapewnianie środków komunikacji między procesami,
- zapewnienie mechanizmów obsługi zakleszczeń (nieobowiązkowe).

Pamięć operacyjna

Pamięć operacyjna stanowi główny magazyn danych dla procesora. Można ją zobrazować, jako tablicę komórek o wielkości 1 bajta (najpopularniejsze rozwiązanie). Każda z tych komórek posiada swój unikatowy adres. Do pamięci operacyjnej bezpośredni[1] dostęp ma procesor oraz urządzenia obsługiwane w trybie DMA. Ponieważ pamięć operacyjna, jak każda inna ma skończoną wielkość, to zarządzanie nią jest ważnym zdaniem systemu operacyjnego. Ma to szczególne znaczenie zwłaszcza w systemach wielozadaniowych.

[1] W przypadku nowszych komputerów to stwierdzenie nie do końca jest prawdziwe, a to za sprawą pamięci podręcznej (ang. cache).

Obsługa pamięci operacyjnej

System operacyjny wykonuje następujące czynności w stosunku do pamięci operacyjnej:

- utrzymuje ewidencję obszarów pamięci, które są w danej chwili zajęte, wraz z informacją do kogo one należą,
- decyduje o tym, które procesy zostaną umieszczone w wolnych obszarach pamięci,
- przydziela i zwalnia obszary pamięci, w zależności od zapotrzebowania.

Pamięć pomocnicza

Pamięć pomocnicza (ang. *external memory*) realizowana jest w postaci pamięci masowej i stanowi uzupełnienie pamięci operacyjnej, która może się okazać niewystarczająca dla procesów użytkownika. Ponieważ urządzenia pamięci masowej są na ogół wolniejsze od pamięci RAM, to konieczne jest efektywne zarządzanie pamięcią pomocniczą.

Zarządzanie pamięcią pomocniczą

Do obowiązków systemu operacyjnego, jako zarządcy pamięci pomocniczej należy:

- zarządzanie obszarami wolnymi,
- przydzielanie obszarów pamięci pomocniczej.

System wejścia-wyjścia

Jednym z naczelnych zadań systemu operacyjnego jest ochrona urządzeń peryferyjnych przed nieprawidłowym ich użyciem przez procesy użytkownika. Efektem tej ochrony jest ukrycie przed procesami użytkownika szczegółów obsługi tych urządzeń. Ma to dodatkową zaletę - zwiększa elastyczność systemu. Opisany na poprzednim wykładzie system przerwań pozwala skonstruować wydajny *system wejścia-wyjścia*. Większość ze współczesnych systemów operacyjnych łączy obsługę urządzeń zewnętrznych z obsługą plików.

Zarządzanie urządzeniami wejścia-wyjścia

System operacyjny tworzy warstwę abstrakcji ułatwiającą procesom użytkownika korzystanie z urządzeń peryferyjnych, która może składać się z np.:

- systemu buforowo-notatnikowego,
- interfejsu do podprogramów obsługi urządzeń peryferyjnych,
- podprogramów obsługi urządzeń peryferyjnych.

Systemy plików

Zawartość pamięci operacyjnej jest ulotna, tzn. przestaje istnieć wraz z wyłączeniem zasilania. Ważne informacje, w tym dane i programy powinny więc zostać zapamiętane na nośnikach, które pozwalają je przechować w sposób trwały. Istnieje wiele urządzeń, które mogą służyć jako pamięć masowa. Każde z tych urządzeń ma specyficzną budowę i sposób obsługi. Aby ujednolicić dla procesów użytkownika sposób korzystania z tych urządzeń system operacyjny tworzy *system plików*. Plik jest jednostką informacji, do której dostęp nie jest zależny do specyfiki nośnika na którym jest przechowywana. Struktura plików zależy od ich twórców. Do przechowywania informacji o plikach i ich porządkowania służą *katalogi*. Niektóre systemy operacyjne implementują katalogi jako specjalny rodzaj plików (pliki gromadzące informacje — metadane — o innych plikach).

Zarządzanie plikami i katalogami

System operacyjny nie tylko *tworzy* system plików, ale również jest odpowiedzialny za:

- tworzenie i usuwanie plików oraz katalogów,
- dostarczanie podstawowych operacji do manipulowania plikami i katalogami,
- odwzorowywanie całości, lub części plików w pamięci operacyjnej,
- umieszczenie plików w pamięci trwałej (masowej).

Sieć

Sieci komputerowe (ang. *networks*) służą do komunikacji pomiędzy systemami komputerowymi i umożliwiają budowę tzw. systemów rozproszonych. Sieci mogą mieć różny zasięg i różne topologie. Systemy komputerowe połączone w sieć mogą być jednakowego typu (sieć homogeniczna) lub różnych typów (sieć heterogeniczna).

Obsługa sieci

W ramach obsługi sieci system operacyjnych może wykonywać następujące czynności:

- wyznaczanie tras pakietów,
- translacja nazw komputerów połączonych w sieć na ich adresy,
- dzielenie i scalanie pakietów,
- nawiązywanie i kończenie połączeń,
- obsługa błędów transmisji.

System ochrony

Ochrona nie jest jednym spójnym podsystemem, ale paradoksalnie jest niezbędna do zapewnienia spójności i stabilności działania systemu komputerowego. W skład tego „podsystemu” wchodzą mechanizmy pozwalające wykrywać próby nieupoważnionego dostępu do zasobów oraz im zapobiegać.

Usługi systemu operacyjnego

Obok zarządzania zasobami i nadzoru nad procesami system operacyjny dostarcza zarówno procesom użytkowników określonych usług. Dzięki tym usługom tworzy środowisko w którym mogą się wykonywać te procesy. To jakie usługi i w jaki sposób dostarcza system operacyjny zależy od wielu czynników, niemniej można wyróżnić kilka grup usług, które są świadczone przez prawie każdy system operacyjny.

Wykonanie programu

Na życzenie użytkownika system operacyjny powinien załadować określony program do pamięci i umożliwić mu jego wykonanie. Program powinien móc zasygnalizować stan swojego wykonania systemowi operacyjnemu (poprawny/niepoprawny).

Operacje wejścia-wyjścia

Procesy użytkownika nie powinny mieć możliwości używania urządzeń peryferyjnych bezpośrednio, bo mogłoby to prowadzić do szeregu nadużyć z ich strony. Opracowywanie fragmentów kodu związanego z wejściem-wyjściem byłoby również uciążliwe dla programistów piszących aplikacje. Dlatego to system operacyjny jest wyposażony w odpowiednie elementy umożliwiające procesom użytkownika wykonanie rozważanych operacji.

Manipulowanie systemem plików.

Ponieważ pliki są strukturami tworzonymi przez system operacyjny, to również za jego pośrednictwem muszą być obsługiwane. Usługi związane z plikami obejmują ich tworzenie, usuwanie, otwieranie, odczyt, zapis, jak również przemieszczanie i kopiowanie.

Komunikacja

Możemy wyróżnić dwie kategorie sposobów komunikowania się procesów: *komunikację lokalną* i *komunikację sieciową*. Pierwszy rodzaj komunikacji obejmuje wymianę danych za pomocą lokalnych łączy lub za pomocą *pamięci dzielonej*. Wszystkie te środki łączności są zapewniane przez system operacyjny.

Wykrywanie wyjątków

Podczas przetwarzania informacji mogą pojawić się wyjątki. Ich źródłem mogą być nie tylko procesy użytkownika, ale również inne elementy systemu komputerowego. System operacyjny musi zagwarantować wykrywanie wszystkich wyjątków niskopoziomowych i poprawną reakcję na nie.

Przydział zasobów

Każdy proces do wykonania potrzebuje zasobów. W każdym systemie komputerowym występuje ograniczona liczba zasobów. Zarządzanie zasobami staje się szczególnie ważne w systemach wielozadaniowych i wielodostępnych, gdyż od niego zależy efektywność i wygoda używania komputera. Przydział niektórych rodzajów zasobów może być oprogramowany za pomocą dosyć ogólnego kodu, natomiast przydziały innych rodzajów zasobów będą wymagały szczególnych rozwiązań.

Rozliczanie/Profilowanie

Czas pracy pierwszych systemów komputerowych był cenny, ze względu na wartość materialną tych urządzeń. Należało więc starannie mierzyć czas poświęcony na wykonanie przez system zadania użytkownika, aby móc później przedstawić mu wiarygodny rachunek. Z czasem obowiązek dokonywania pomiaru czasu pracy procesów przejęły systemy operacyjne. W nowszych ich wersjach takie usługi są rzadziej spotykane, ale dały one początek usługom, które pozwalają sporządzać statystyki wykorzystania zasobów komputera i tym samym pozwalają na wprowadzenie do systemu poprawek optymalizacyjnych.

Bezpieczeństwo

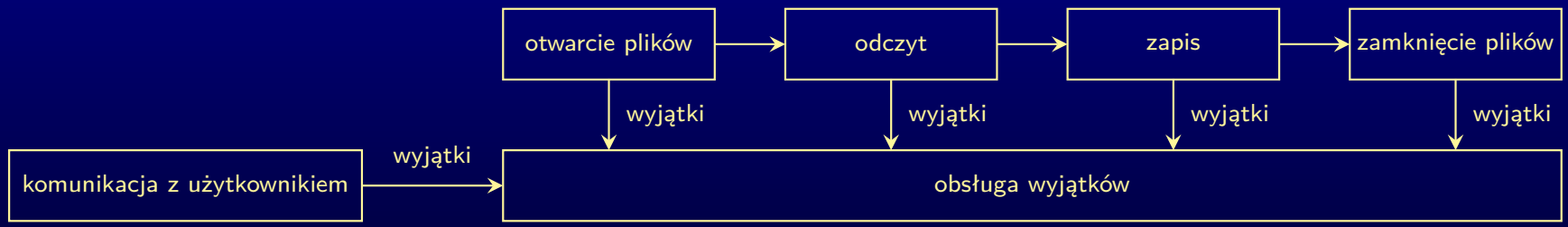
System operacyjny powinien dostarczać swym użytkownikom usług i mechanizmów pozwalających na realizację przyjętej przez nich polityki bezpieczeństwa. Do tych mechanizmów i usług należy zaliczyć zarządzanie prawami dostępu, system uwierzytelniania użytkowników, system rejestrowania zdarzeń zachodzących w systemie, itd.

Wywołania systemowe

Wywołania systemowe (ang. *system calls*) nazywane również *funkcjami systemowymi* są specjalnymi podprogramami (lub podprogramem) systemu operacyjnego, których zadaniem jest obsługa wyróżnionych przerwań (lub przerwania), które pozwalają na komunikację między procesami użytkownika, a systemem operacyjnym. Za pomocą wywołań systemowych procesy użytkownika mogą przedstawiać swe potrzeby systemowi operacyjnemu, a więc tworzą one *interfejs* między tymi dwoma elementami systemu komputerowego. Wywołania systemowe są bezpośrednio dostępne dla programistów piszących aplikacje w języku asemblera oraz w niektórych językach wysokiego poziomu (np. C). Częściej jednak są one wywoływane w sposób pośredni, tzn. języki programowania wysokiego poziomu dostarczają bibliotek podprogramów, które stanowią mniej lub bardziej złożone „opakowania" na wywołania systemowe (np. funkcja `printf()` w języku C).

Przykład

Poniżej przedstawiony jest diagram, który obrazuje z jakich wywołań systemowych może korzystać program kopiujący pliki.



Argumenty wywołań systemowych

Podobnie jak zwykłe podprogramy wywołania systemowe mogą wymagać pewnych argumentów wywołania. Te argumenty mogą być im przekazywane na trzy różne sposoby:

1. przez rejestry,
2. przez stos,
3. przez pamięć — adres początku obszaru pamięci zawierającego argumenty umieszczany jest w rejestrach.

Kategorie wywołań systemowych

Liczba i sposób działania wywołań systemowych jest zależna od usług, jakich system operacyjny dostarcza procesom i użytkownikom. Możemy w związku z tym podzielić funkcje systemowe na kilka kategorii:

1. wywołania związane z zarządzaniem procesami (tworzenie procesu, ładowanie programu do pamięci, kończenie procesu, profilowanie, debugowanie, zawieszanie działania i synchronizacja),
2. wywołania związane z operacjami na plikach (tworzenie, otwarcie, odczyt, zapis, zmiana położenia wskaźnika pliku, zamknięcie),
3. wywołania związane z operacjami na urządzeniach peryferyjnych (głównie te same co dla plików),
4. wywołania związane z utrzymywaniem informacji (odczyt czasu i daty, udostępnianie danych statystycznych, jak ilość wolnego miejsca w pamięci operacyjnej lub masowej, liczba zalogowanych użytkowników, itd.),
5. wywołania związane z komunikacją (tworzenie i usuwanie łącz, nadawanie i odbieranie komunikatów, tworzenie pamięci dzielonej).

Programy systemowe

Wraz z niemalże każdym systemem operacyjnym dostarczane są programy, które nie stanowią części jądra systemu i działają w trybie użytkownika, ale na tyle blisko współpracują z systemem operacyjnym, że można je zaliczyć do oprogramowania systemowego. Najważniejszym przedstawicielem tej grupy jest *interpreter poleceń*. Jego głównym zadaniem jest umożliwienie komunikacji między użytkownikiem komputera, a systemem operacyjnym. Można wyróżnić co najmniej dwa rodzaje takich programów:

1. interpretery tekstowe,
2. interpretery graficzne.

Interpretery tekstowe

Interpretery pracujące w środowisku tekstowym pozwalają komunikować się użytkownikowi z komputerem za pomocą *wiersza poleceń* (ang. *command line*). Prostym przykładem takiego interpretera jest `commmand.com` z systemu MS-DOS. Ładując do pamięci program, który mu został przedłożony do wykonania, nie tworzy on nowego procesu lecz usuwa fragment siebie, zwalniając tym samym część pamięci operacyjnej, którą przeznacza dla programu użytkownika. Po zakończeniu wykonania programu, sterowanie wraca do interpretera, który odbudowuje się. Bardziej wyrafinowaną postacią interpreterów poleceń są powłoki (ang. *shell*) w systemie Unix (`bash`,`tcsh`,`zsh`). Są one wykonywane jako osobne procesy. Kiedy muszą wykonać inny proces, to tworzą proces potomny, którego program jest zastępowany zleconym zadaniem. W zależności od sposobu uruchomienia nowego procesu sterowanie może wrócić natychmiast do interpretera, lub po zakończeniu procesu. Polecenia powłoki mogą stanowić część jej kodu, bądź być osobnymi programami.

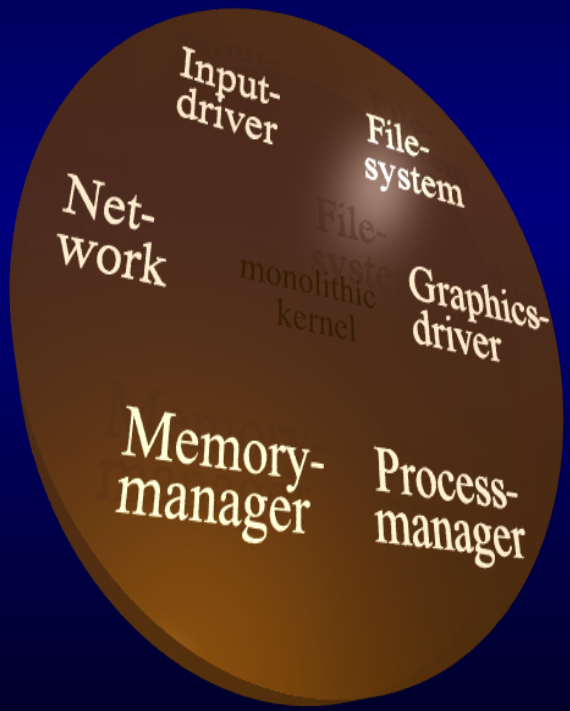
Interpretery graficzne

Interpretery graficzne wyposażone są w *graficzny interfejs użytkownika* (GUI), pozwalający porozumiewać się z komputerem za pomocą wskaźnika myszy oraz systemu okien i ikon. Interpretery te mogą być na stałe zintegrowane z systemem operacyjnym (GUI systemów MacOS, explorer w Windows) lub być osobnym rozbudowanym systemem, takimi jak X Window lub Wayland, które umożliwiają nawet zdalną pracę przez sieć i zmianę interfejsu użytkownika, od prostych zarządców okien (i3, Enlightenment, Window Maker), po całe środowiska graficzne (Gnome, KDE). Pojawiły się również próby wykorzystania w interpreterach graficznych grafiki 3D (Looking Glass, Compiz, Beryl, Areo).

Struktury jądra systemu operacyjnego

Ponieważ napisanie jądra systemu operacyjnego jest złożonym przedsięwzięciem, to musi je poprzedzić proces przygotowań w wyniku którego zostają podjęte decyzje, co do struktury i funkcjonowania jądra. Na następnych planszach zostaną przedstawione ogólne rozwiązania takiego problemu.

Jądro monolityczne



Jądro monolityczne jest jednym programem, podzielonym na pod-programy, które wzajemnie są ze sobą powiązane. Brak w nich wyraźnej struktury, lub jest ona dosyć luźna. Przykłady: Unix, MS-DOS,MS-Windows 95,98,ME.

Jądro systemu MS-DOS

W systemie MS-DOS, procesy użytkownika (a w zasadzie jeden proces) mogą korzystać zarówno z funkcji dostarczanych przez system DOS, jak i z usług dostarczanych przez rezydentne programy systemowe oraz BIOS. Mogą uzyskiwać również bezpośredni dostęp do sprzętu.

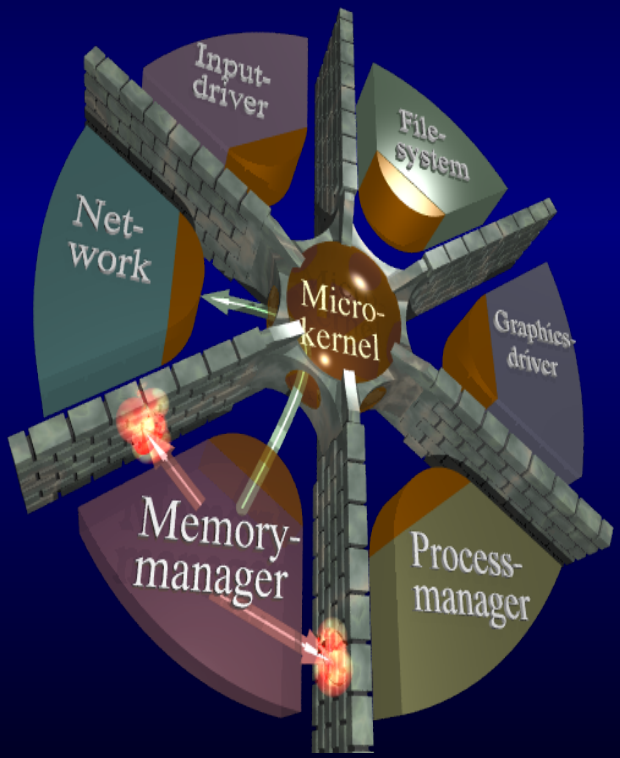
Jądro systemu Unix

Oryginalne jądro systemu Unix było zaprojektowane dla sprzętu nieposiadającego żadnego mechanizmu ochrony. Mimo to twórcy systemu postanowili dokładnie odseparować procesy użytkownika od sprzętu i umożliwić dostęp do niego tylko za pomocą wywołań systemowych.

Jądro monolityczne z modułami

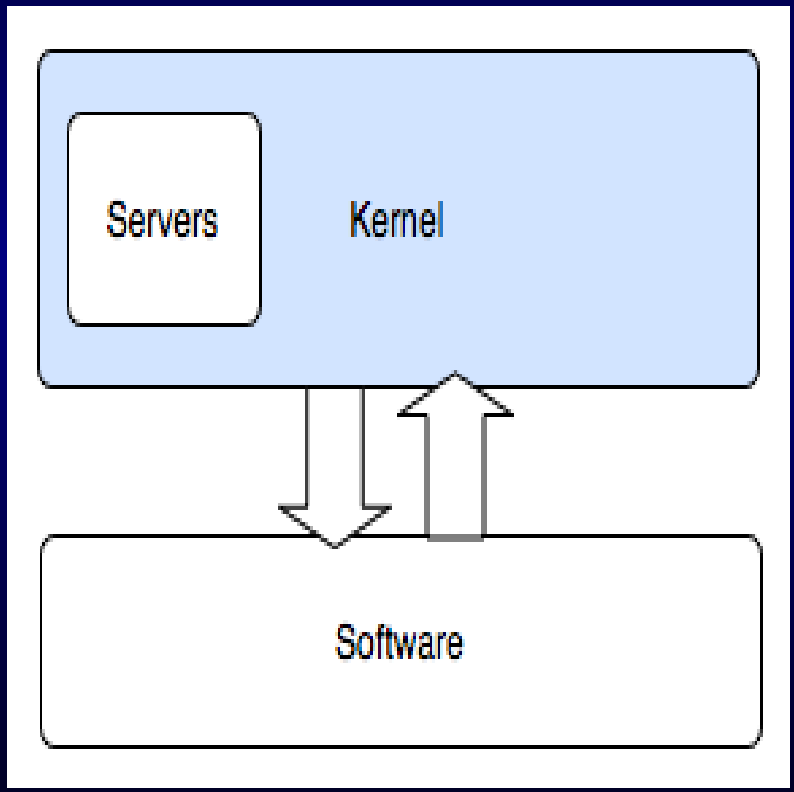
Jest to pewna modyfikacja jądra monolitycznego, pozwalająca na ładowanie w trakcie działania jądra pewnych jego fragmentów (np.: sterowników urządzeń) do pamięci, na podobnej zasadzie, jak programy użytkowników ładują biblioteki współdzielone. Jądro takie musi być wyposażone w dodatkowe elementy: tablicę symboli, mechanizm ładowania modułu i mechanizm śledzenia zależności między modułami. Przykładami systemów z jądrem modularnym są Linux i FreeBSD.

Mikrojądro (ang. microkernel)



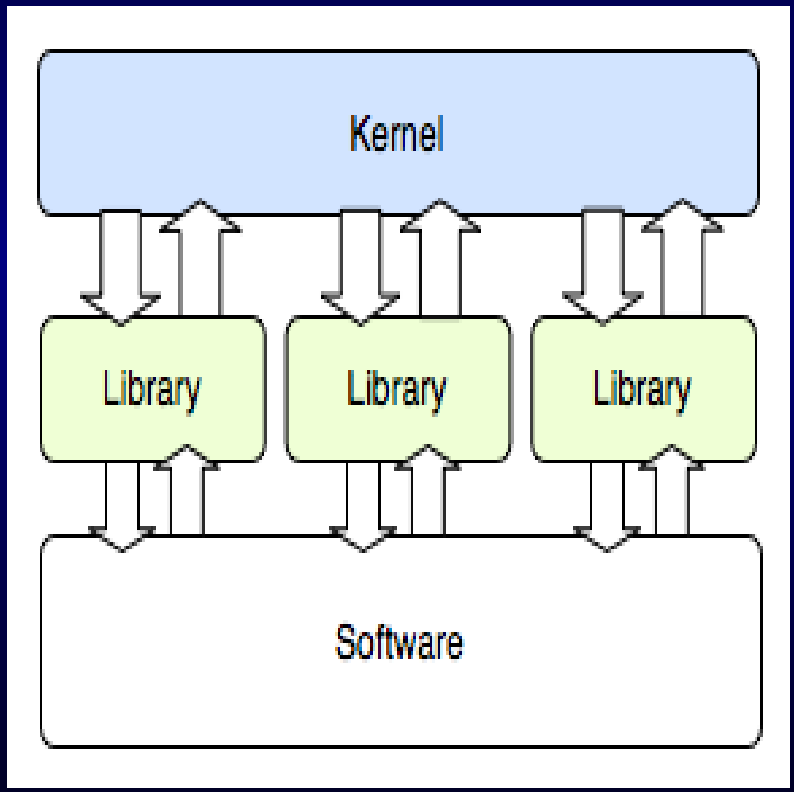
Jądro ma niewielkie rozmiary i jest wykonywane w trybie systemowym procesora. Zadania jądra sprowadzają się do zarządzania procesami, pamięcią operacyjną i zapewnienia komunikacji międzyprocesowej. Inne czynności wykonywane tradycyjnie przez jądro przejmują programy-demony pracujące w trybie użytkownika, lub pośrednim.

Jądro hybrydowe



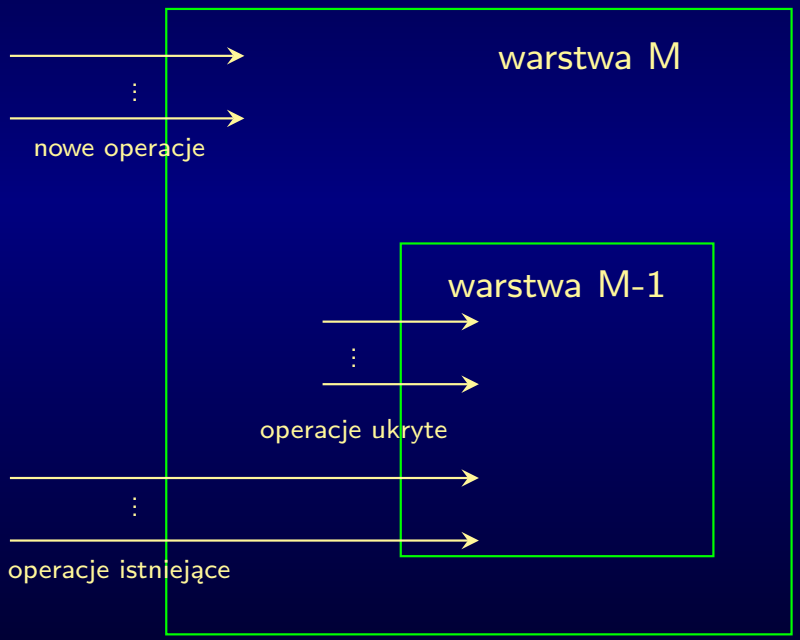
Jest to rozwiązanie pośrednie między jądrem monolitycznym, a mikrojądrem. Wszystkie serwisy są uruchamiane w trybie jądra. Część ekspertów uważa, że taka kategoria nie istnieje naprawdę i jest tylko chwytem marketingowym, a jądro hybrydowe jest zwykłym jądrem monolitycznym, z dobrze określoną strukturą wewnętrzną. Przykłady: Windows NT, 2000, XP.

Egzojądro (ang. exokernel)



Egzojądro (ang. *exokernel*) jest stosunkowo nową koncepcją w dziedzinie systemów operacyjnych. Jądro jest jeszcze mniejszych rozmiarów niż w przypadku mikrojądra. Jego jedynym zadaniem jest zapewnienie ochrony zasobów na niskim poziomie. Wszelkie rodzaje abstrakcji, takie jak np.: system plików są dostarczane procesom użytkownika za pomocą zewnętrznych bibliotek. Podobne rozwiązania noszą nazwy pikojąder i nanojąder.

Jądro z podziałem na warstwy.



Jądro systemu jest podzielone na współpracujące ze sobą warstwy. Każda warstwa posiada określony interfejs, który udostępnia warstwie następnej. Ta z kolei może udostępniać bezpośrednio część funkcji z warstwy poprzedniej, warstwie która znajduje się nad nią, może ukrywać część funkcji z warstwy poprzedniej i może definiować własne funkcje. Przykłady: THE, Venus, MULTICS.

Maszyny wirtualne

Maszyny wirtualne są naturalnym rozwinięciem idei podziału jądra na warstwy. Taka maszyna jest po prostu wirtualną kopią komputera, którą otrzymuje każdy z uruchomionych programów. Maszyna ta może pracować w rzeczywistym trybie użytkownika i wykonywać system operacyjny, pracujący w wirtualnym trybie systemowym, który będzie wykonywał procesy działające w wirtualnym trybie użytkownika. Jednym z pierwszych systemów wykorzystujących koncepcję maszyn wirtualnych był IBM VM/370. Obecnie ta idea zyskuje na popularności dzięki pojawieniu się w najnowszych procesorach sprzętowego wsparcia dla wirtualizacji.

Projekt systemu operacyjnego

Podobnie jak w przypadku innych dużych projektów, nie ma gotowych przepisów na napisanie systemu operacyjnego. Projekt takiego oprogramowania jest kompromisem między wymaganiami narzucanymi przez użytkowników (wygoda obsługi), programistów (wygoda tworzenia nowych aplikacji, utrzymania (ang. maintenance)) i sprzęt (dostępne urządzenia, mechanizmy ochrony). Również jak w przypadku innego oprogramowania przydatne są zasady inżynierii oprogramowania (np.: podział projektu na mniejsze części) i mniej oficjalne reguły (KISS — Keep It Simple, Stupid, DRY — Don't Repeat Yourself).

Polityka i mechanizmy

Podstawową zasadą przy tworzeniu systemów operacyjnych jest oddzielenie mechanizmu od polityki. Polityka określa *co* ma być zrobione, natomiast mechanizm określa *w jaki sposób*. Mechanizmy powinny być na tyle elastyczne, aby pozwalały zrealizować dowolną politykę. Twórca systemu nie powinien również narzucać jego użytkownikom (głównie administratorom) żadnej polityki. Przykład: w systemie trzeba uwierzytelniać użytkowników (polityka), może to się odbywać za pomocą haseł statycznych, zmiennych w czasie lub za pomocą systemów biometrycznych (mechanizmy).

Języki programowania

Początkowo wszystkie systemy operacyjne były pisane w języku asemblera. Z czasem pojawiły się języki wysokiego poziomu, których można było użyć zamiast języków niskopoziomowych, takimi językami były Bliss, Concurrent Pascal, niektóre dialekty języka Fortran. Prawdziwym przełomem było pojawienie się języka C, który został opracowany specjalnie na potrzeby prac nad systemem Unix. Większość współczesnych systemów jest napisana w tym języku. Obecnie używane są również C++ (Haiku, UnixLite), Java (JavaOS, JNode), C# (Singularity, OSharp, Cosmos) i Rust (Redox OS).

Instalacja

Istnieją dwa skrajne scenariusze instalowania systemów operacyjnych: kompilacja całości systemu, dzięki czemu otrzymujemy system całkowicie dostosowany do sprzętu którym dysponujemy i do naszych potrzeb oraz instalacja wersji binarnej, co jest niewątpliwie szybszym rozwiązaniem. Rozwiązanie pośrednie polega na zainstalowaniu części binarnej i kompilacji elementów systemu, które mogą mieć wpływ na efektywność korzystania z systemu komputerowego.

Pytania

?

Źródła

Rysunki jądra monolitycznego i mikrojądra pochodzą ze strony TU Dresen: Operating System Group. Schematy jądra hybrydowego i egzojądra pochodzą ze stron Wikipedii i są chronione przez licencję Creative Commons.

Koniec

Dziękuję Państwu za uwagę!