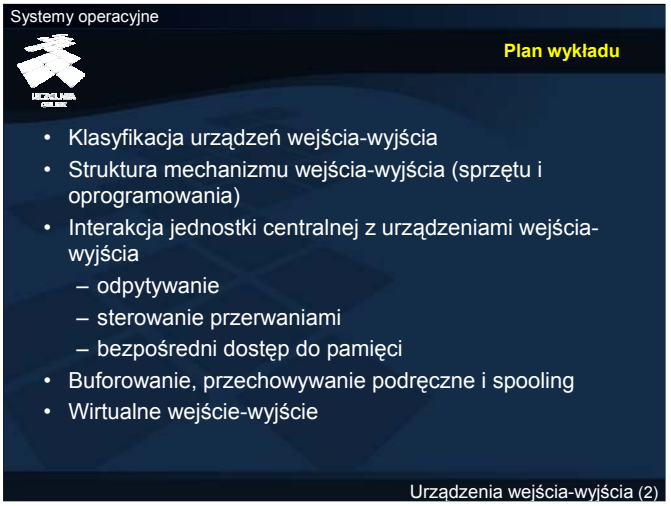
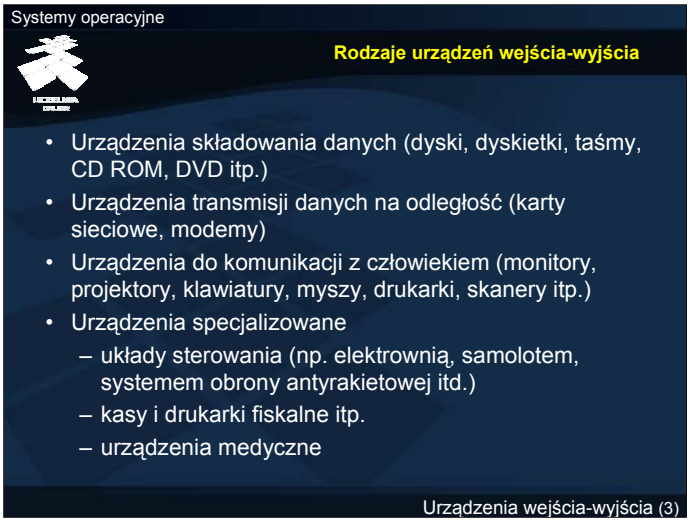
Celem wykładu jest omówienie zagadnień obsługi urządzeń wejścia-wyjścia (zwanych również urządzeniami zewnętrznymi lub peryferyjnymi) i realizacji związanych z tym mechanizmów w jądrze systemu operacyjnego. Problem obsługi urządzeń wejścia-wyjścia jest o tyle skomplikowany, że są to urządzenia bardzo zróżnicowane pod wieloma względami, stosunkowo wolne (w porównaniu z jednostką centralną) i stanowią najczęściej zmieniający się element konfiguracji systemu komputerowego. Z drugiej strony urządzenia wejścia-wyjścia stanowią „zmysły” komputera, dlatego większość z nich jest bardzo istotna dla użytkownika i jego interakcji z systemem. Można wręcz powiedzieć, że zwykły użytkownik postrzega komputer właśnie poprzez urządzenia wejścia-wyjścia. Efektywność i wygoda obsługi tych urządzeń, zwłaszcza w systemach interaktywnych, decyduje więc o ogólnym wrażeniu z jakości pracy z komputerem.



Treść wykładu obejmuje:

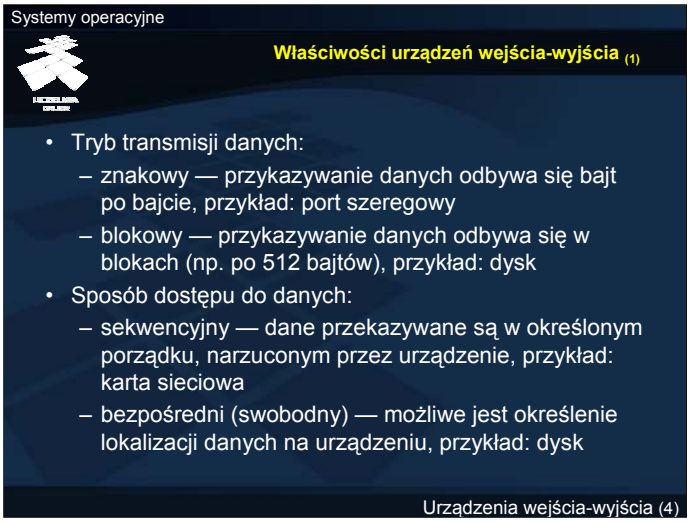
- klasyfikację urządzeń wejścia-wyjścia według różnych kryteriów,
- przedstawienie ogólnej struktury mechanizmu wejścia-wyjścia z krótkim odniesieniem do kluczowych kwestii sprzętowych,
- omówienie sposobów interakcji jednostki centralnej z urządzeniami wejścia-wyjścia wraz z dyskusją zagadnień efektywności,
- przedstawienie technik poprawy efektywności interakcji jednostki centralnej z urządzeniami wejścia-wyjścia, opartych na różnych formach buforowaniu,
- odniesienie do wirtualnego wyjścia-wyjścia, tworzonego przez jądro na bazie urządzeń fizycznych w celu ułatwienia wykorzystania ich zasobów i możliwości.



Urządzenia wejścia-wyjścia są bardzo zróżnicowane pod każdym względem. Przedstawione wyszczególnienie dotyczy zastosowań. Warto podkreślić, że urządzenia składowania danych sterowane są wyłącznie przez jednostkę centralną. Są więc właściwie niezależne od zdarzeń poza systemem komputerowym, z wyjątkiem szczególnych przypadku wyjęcia dyskietki, płyty CD itp.

Urządzenia transmisji na odległość oprócz reakcji na sygnały sterujące ze strony jednostki centralnej reagują również na zdarzenia zewnętrzne, związane z przekazywaniem danych z innych jednostek. Należy tu też podkreślić, że każda komunikacja z urządzaniem zewnętrznym jest jakąś transmisją danych. Urządzenia transmisji danych na odległość służą do wymiany danych z innymi komputerami, więc chodzi tu o transmisję danych pomiędzy urządzeniami o podobnym charakterze, a nie transmisję pomiędzy jednostką centralną, a zintegrowanym z nią urządzeniem.

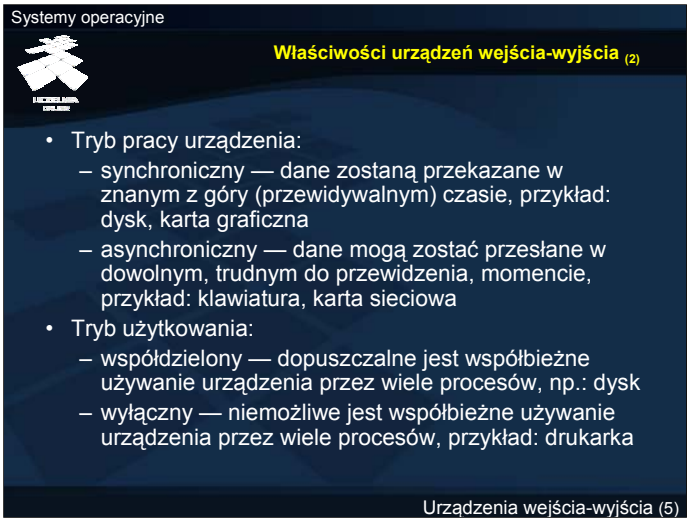
Urządzenia do komunikacji z człowiekiem w ogólności też reagują zarówno na zdarzenia wewnętrzne, jak i zewnętrzne. Zdarzenia zewnętrzne dotyczą przede wszystkim urządzeń wejściowych i związane są z działaniami człowieka (użytkownika). Są to więc zdarzenia, które zachodzą niezbyt często w porównaniu z szybkością pracy jednostki centralnej, a moment ich zajścia jest trudny do przewidzenia.



Zróżnicowanie urządzeń wejścia-wyjścia przejawia się między innymi w dużej liczbie klasyfikacji, wynikających z różnych kryteriów.

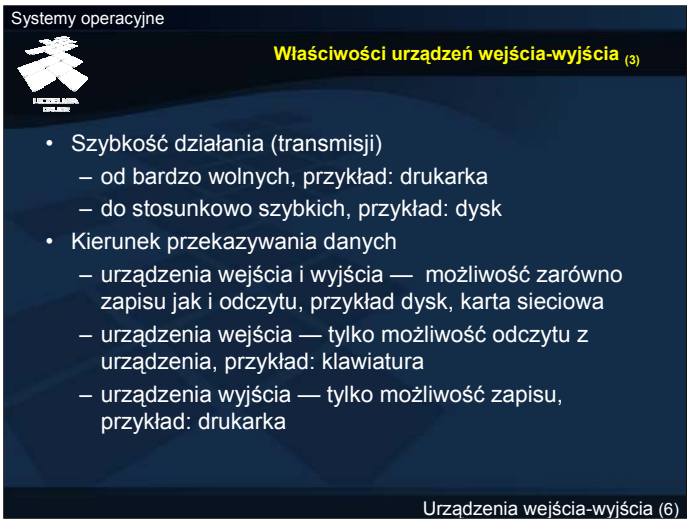
Ze względu na tryb transmisji wyodrębnia się urządzenie znakowe i blokowe. W przypadku urządzeń blokowych w wyniku operacji wejścia-wyjścia następuje przekazanie bloku o ustalonym rozmiarze np. 512 B. W przypadku urządzeń znakowych po przekazaniu bajta lub słowa konieczne jest zlecenie następnej operacji wejścia-wyjścia w celu odebrania kolejnej takiej jednostki. Urządzenia takie mają czasami wewnętrzny bufor, umożliwiający przekazywanie nieco większych porcji danych.

Podział na urządzenia sekwencyjne i bezpośrednie wiąże się z łatwością uzyskania dostępu do wybranego zakresu danych na urządzeniu. Większość urządzeń wejścia-wyjścia pracuje w sposób sekwencyjny, czyli przekazuje dane do jednostki centralnej w postaci pewnego strumienia bez możliwości ograniczenia do pewnego zakresu. Jednostka centralna może co najwyżej dokonać filtracji po odebraniu tych danych. Podobnie w przypadku przekazywania danych do urządzenia, nie ma możliwości wskazania kolejności przetwarzania czy zmiany kolejności ułożenia poszczególnych części. Jednym z niewielu urządzeń o dostępie bezpośrednim jest dysk. Przed zainicjalizowaniem właściwej operacji dostępu można odpowiednio ustawić pozycję głowicy oraz wskazać sektor do zapisu lub odczytu. Podobne podejście można by potencjalnie stosować w przypadku napędów taśmowych. Zasadnicza trudność wiąże się jednak z czasem pozycjonowania, wymagającym długotrwałego przewijania taśmy. Dlatego napędy taśmowe są urządzeniami o dostępie sekwencyjnym.



W przypadku urządzeń synchronicznych wymagany czas pracy urządzenia można w miarę precyzyjnie przewidzieć. Taki czas jest jednak zmienna losową, ale o stosunkowo niewielkim rozproszeniu wartości. Na przykładzie dysku można powiedzieć, że po ustawieniu pozycji głowicy sam czas odczytu sektora do się precyzyjnie określić, ale opóźnienie obrotowe ma charakter losowy. Elementem losowym, zwiększającym czas dostępu są również przypadki błędu w odczycie sektora. Precyzyjne przewidywanie nie jest możliwe w przypadku urządzeń „synchronizowanych” zdarzeniami zewnętrznymi. Dla jednostki centralnej zdarzenia te zachodzą całkowicie asynchronicznie, nie da się więc przewidzieć momentu, w którym urządzenie będzie wymagało obsługi ze strony jednostki centralnej.

Pewne urządzenia mogą współbieżnie obsługiwać zlecenia wielu procesów. Dla urządzenie nie ma znaczenia jaki proces czy użytkownika obsługuje. Pojęcie procesu na poziomie architektury nawet nie istnieje. Ma to natomiast znaczenia dla użytkownika. Dla użytkownika nie ma znaczenia, czy dysk zapisując jego dane, wplecie pomiędzy operacje zapisu poszczególnych sektorów, operacje zapisu lub odczytu innych sektorów, wynikające z osobnego zlecenia. Co najwyżej nastąpi opóźnienie realizacji całego zlecenia zapisu, jest to jednak raczej nieodczuwalne. Trudno jednak taki przeplot zaakceptować na wydruku — jest mało prawdopodobne, żeby wydruk był czytelny.



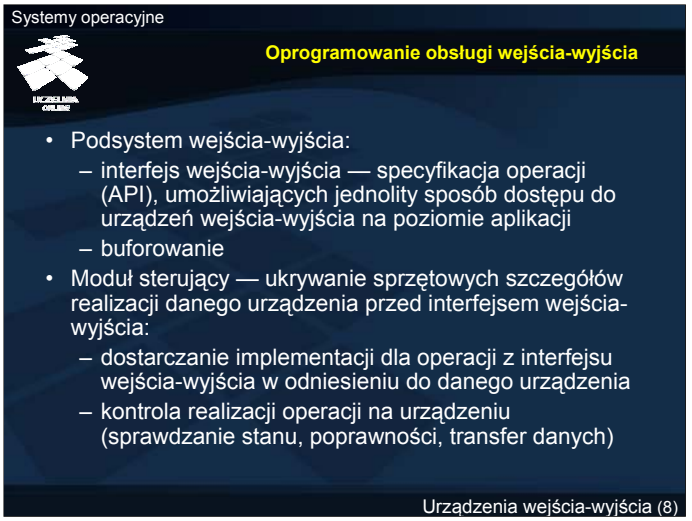
W zakresie szybkości działania nie jest łatwo wyodrębnić konkretne klasy. Celem tej klasyfikacji jest podkreślenie dużego zróżnicowania w zakresie szybkości przetwarzania danych przez urządzenie. Oczywiście wszystkie urządzenia wejścia-wyjścia są wolne w porównaniu z jednostką centralną.

Kierunek przekazywania danych dotyczy danych przetwarzanych, a nie sygnałów sterujących, czy informacji o stanie urządzenia. Drukarka jest typowym urządzeniem wyjściowym, ale można odczytać jej stan (np. informację o braku papieru, tonera itp.). Nie służy ona natomiast do wprowadzania danych.

Chociaż można sobie wyobrazić urządzenie wejścia-wyjścia sterowane bezpośrednio przez procesor główny, współcześnie takich rozwiązań raczej się nie stosuje. Urządzenia mają swoje sterowniki (ang. device controller), czyli układy elektroniczne, odpowiedzialne za kontrolę ich pracy. Dzięki sterownikowi możliwa jest równoległa praca jednostki centralnej i urządzenia wejścia-wyjścia.

Interakcja jednostki centralnej z urządzeniem wejścia-wyjścia sprowadza się do zapisu lub odczytu odpowiednich rejestrów sterownika. Sterownik może być umieszczony na płycie głównej (np. karta graficzna) lub na płytce urządzenia (np. sterownik dysku, sterownik drukarki). Sterownik na płycie głównej lub ta jego część, która przystosowana jest do współpracy z magistralą systemu komputerowego, nazywany jest adapterem. W przypadku niektórych urządzeń komunikacja pomiędzy sterownikiem, a adapterem na płycie odbywa się za pośrednictwem specjalnej magistrali (np. magistrali SCSI). Procesor ma wówczas bezpośredni dostępu do rejestrów adaptera, który komunikuje się ze sterownikiem. W przypadku, kiedy sterownik nie jest zintegrowany z adapterem, komunikacja pomiędzy urządzeniem zewnętrznym a jednostką centralną może odbywać się przez odpowiedni port standardowy, np. port szeregowy (RS-232, USB) lub port równoległy. W tym przypadku procesor również nie ma bezpośredniego dostępu do rejestrów sterownika i wszystkie operacje zapisu oraz odczytu musi wykonać za pośrednictwem sterownika portu i jego rejestrów.

Zadaniem systemu operacyjnego jest między innymi ułatwienie dostępu do urządzeń, przez ujednolicenie i uproszczenie interfejsu, czyli ukrycie szczegółów realizacji urządzenia. Odpowiada za to przede wszystkim moduł sterujący (moduł obsługi urządzenia, ang. device driver).

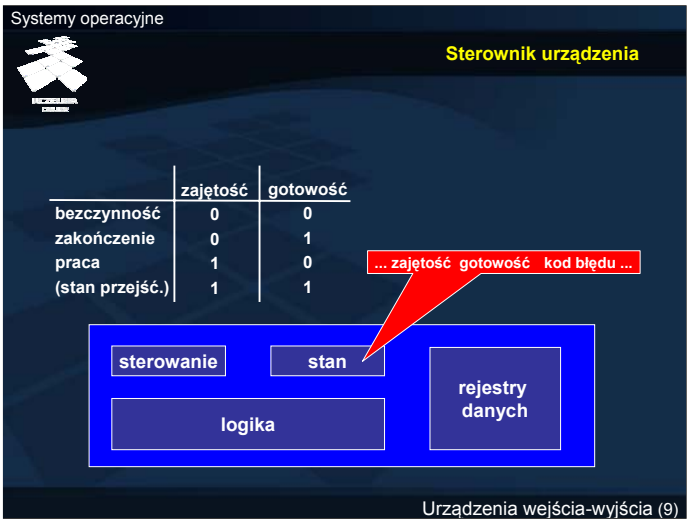


Podsystem wejścia-wyjścia realizuje zadania niezależne od konkretnych urządzeń: dostarcza ogólny interfejs i realizuje buforowanie. Celem projektowym interfejsu wejścia-wyjścia jest dostarczanie aplikacji interfejsu funkcji (API), umożliwiających wykonywanie operacji wejścia-wyjścia w sposób jednolity, niezależny od urządzenia lub grupy, do której należy urządzenie. Typowy interfejs obejmuje między innymi funkcje:

- read — odczyt z urządzenia (pobieranie danych),
- write — zapis do urządzenia (wysyłanie danych).

Sposób implementacji tych funkcji zależy od specyfiki urządzenia. Implementacja taka dostarczana jest przez moduł sterujący. Moduł sterujący dostarcza też procedur, które nie są dostępne w interfejsie dla aplikacji, ale wywoływane są np. w ramach obsługi przerwania, zgłaszanego przez urządzenie.

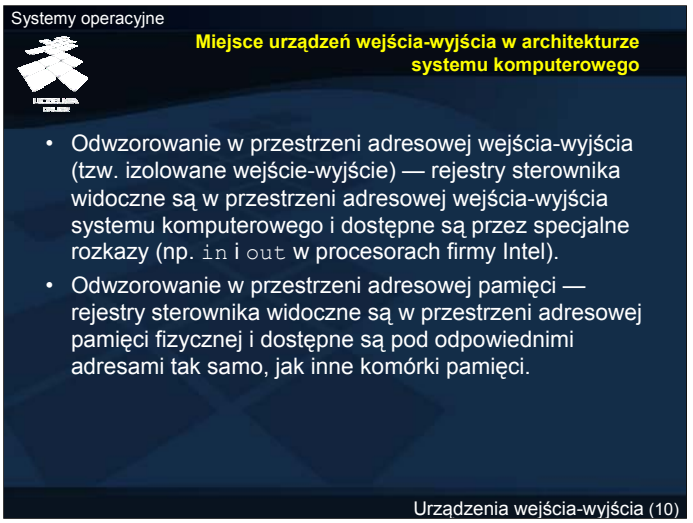
Moduły sterujące dostarczane są dla typowych systemów operacyjnych (np. Windows XP, Solaris, Linux) przez twórców systemów operacyjnych lub wytwórców urządzeń wejścia-wyjścia. Moduł sterujący jest taką częścią oprogramowania systemowego, które może wymagać modyfikacji i uzupełnień, zależnie od urządzeń dołączanych do komputera. Jądro systemu operacyjnego musi więc dostarczyć odpowiednie mechanizmy, umożliwiające dołączanie nowych modułów. W skrajnym przypadku sposobem dołączania nowego modułu jest rekompilacja jądra. Metoda taka stosowana była w klasycznym systemie UNIX, dlatego elementem standardowego oprogramowania tych systemów był między innymi kompilator języka C.



Sterownik urządzenia dostępny jest dla jednostki centralnej poprzez odpowiedni zbiór rejestrów. Procesor ma bezpośredni dostęp do rejestrów tych sterowników, które podłączone są do magistrali systemowej. Oznacza to, że w niektórych przypadkach procesor ma dostęp tylko do rejestrów sterownika portu, do którego podłączone jest urządzenie. Przez ten port następuje wymiana informacji pomiędzy jednostką centralną a właściwym sterownikiem urządzenia.

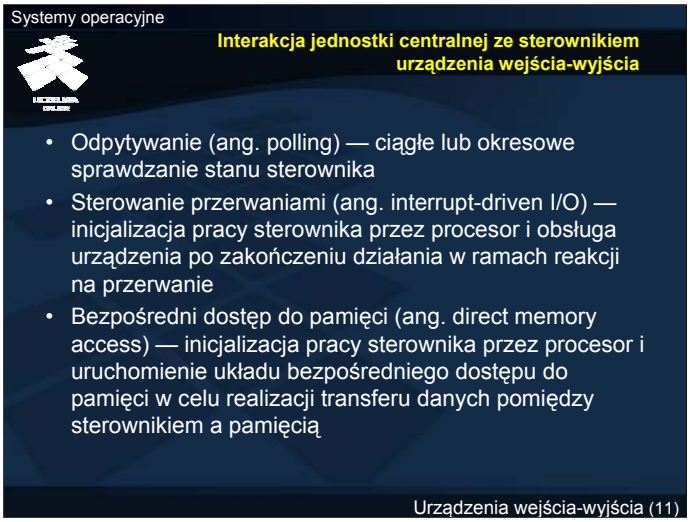
Liczba i wielkość rejestrów sterownika zależą od konkretnych rozwiązań. W typowym sterowniku można jednak wyróżnić następujące rejestry:

- Rejestr stanu (ang. status register) — jest czytany przez procesor i zawiera bity określające stan sterownika np.:
  - bit *gotowości* — sygnalizujący zakończenie zlecenia i gotowość przekazania danych lub informacji o błędzie (bit gotowości może być automatycznie kasowany po odczytaniu danych lub informacji o błędzie),
  - bit *zajętości* — sygnalizujący prace urządzenia (realizację operacji wejścia-wyjścia),
  - bity kodu błędu — sygnalizujące przyczynę niepowodzenia operacji.
- Rejestr sterowania (ang. control register, command register) — zawiera bity definiujące tryb pracy urządzenia, rozpoczęcie realizacji polecenia itp. Rejestr jest najczęściej zapisywany przez procesor.
- Rejestr danych wejściowych (ang. data-in register) — jest czytany przez procesor w celu odbioru danych z urządzenia.
- Rejestr danych wyjściowych (ang. data-out register) — jest zapisywany przez procesor w celu wysłania danych do urządzenia.



Różnica pomiędzy odwzorowaniem rejestrów w przestrzeni pamięci i w przestrzeni wejścia-wyjścia polega na przekazaniu do odpowiedniego dekodera adresowego innych sygnałów sterujących (podłączeniu innych linii magistrali sterującej). W przypadku odwzorowania w pamięci przekazywane są sygnały zapisu/odczytu pamięci, a w przypadku odwzorowania w przestrzeni wejścia-wyjścia przekazywane są sygnały zapisu/odczytu wejścia-wyjścia.

Niektóre urządzenia udostępniają część swoich rejestrów w przestrzeni adresowej wejścia-wyjścia, a część w przestrzeni adresowej pamięci (np. karta graficzna).



Zasadniczą kwestią w interakcji z urządzeniem zewnętrznym jest sposób przekazywania informacji o stanie urządzenia pomiędzy procesorem a sterownikiem oraz danych pomiędzy sterownikiem a pamięcią.

W przypadku odpytywania procesor jest odpowiedzialny zarówno za monitorowanie stanu sterownika (np. w celu stwierdzenia zakończenia operacji) jak i transfer danych. Procesor jest więc zobligowany do ciągłego lub okresowego sprawdzania rejestru stanu sterownika, co wymaga odpowiedniej konstrukcji modułu sterującego. Podejście tego typu określa się jako *aktywne czekanie*. Odpytywanie może być stosowane w przypadku urządzeń synchronicznych, wykonujących krótkotrwałe operacje wejścia-wyjścia.

W przypadku sterowania przerwaniami procesor jest odpowiedzialny za transfer danych, ale nie musi monitorować w sposób ciągły stanu sterownika. Inicjalizuje on pracę sterownika a o jej zakończeniu lub zaistnieniu określonego stanu informowany jest przez przerwanie, które zgłasza sterownik. W oprogramowaniu systemowym należy zatem uwzględnić procedurę obsługi przerwania a jej adres umieścić na właściwej pozycji wektora przerwań.

W przypadku zastosowania układu DMA, po zainicjalizowaniu pracy urządzenia przez procesor, przekazywanie danych pomiędzy sterownikiem a pamięcią realizowane jest przez specjalizowany układ (DMA), który wykonuje swoje zadanie bez angażowania procesora. Zależnie od architektury, zadanie takie może również wykonywać procesor wejścia-wyjścia, który może nawet dysponować własną, prywatną pamięcią.

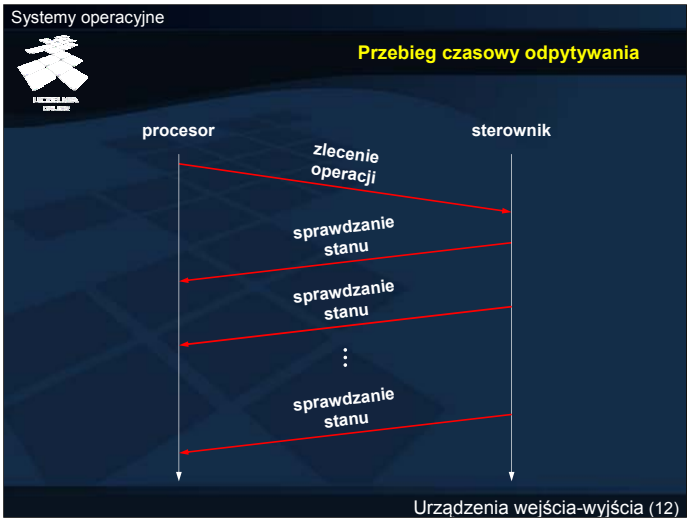
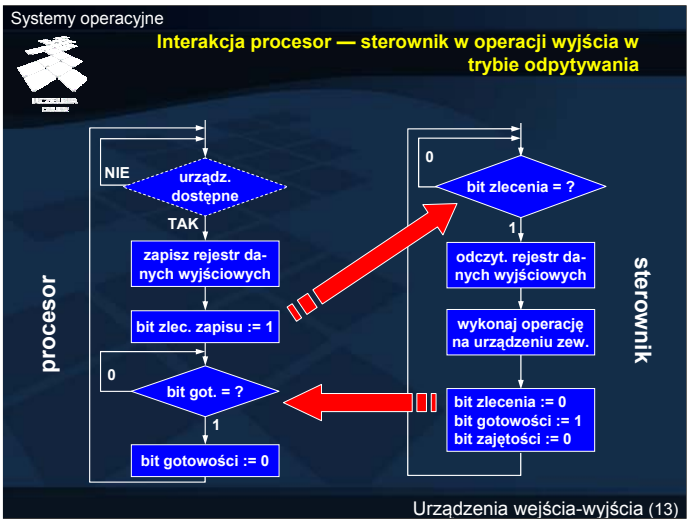


Diagram pokazuje przebieg czasowy interakcji procesora ze sterownikiem w trybie odpytywania. Po zainicjalizowaniu operacji wejścia-wyjścia procesor wielokrotnie odczytuje rejestr stanu sterownika, zanim nastąpi stwierdzenie zakończenia operacji.

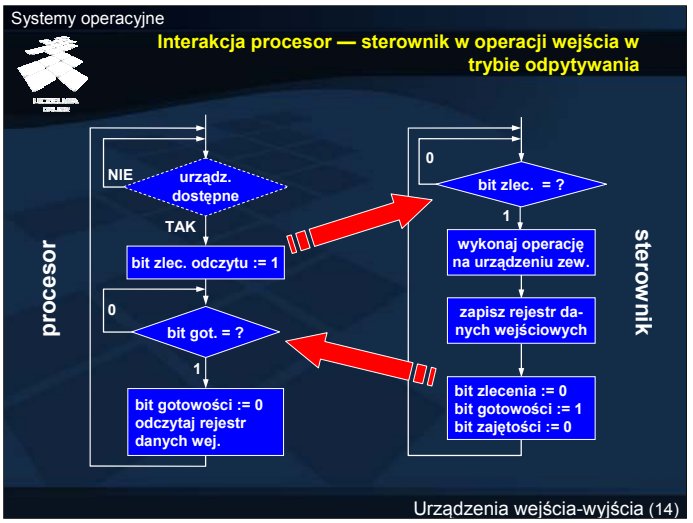


Interakcja pomiędzy procesorem a sterownikiem w najprostszym przypadku mogłaby się opierać na 2 bitach. W zaprezentowanym rozwiązaniu sam stan sterownika reprezentowany jest przez 2 bity (gotowości i zajętości).

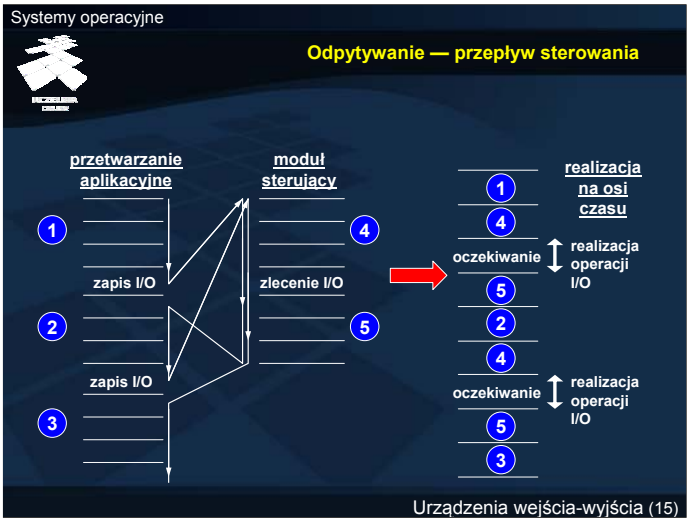
Dla uproszenia dalszej prezentacji przyjęto, że ustawienie bitu gotowości po zakończeniu operacji oznacza pomyślne zakończenie. Brak ustawienia bitu gotowości oznacza błąd, co z kolei jest sygnalizowane przez odpowiednie bity kodu błędu w rejestrze stanu. W przypadku operacji wyjścia bit gotowości oznacza zatem, że udało się przekazać zawartość rejestru danych do urządzenia i ją przetworzyć.

Sterownik urządzenia czeka na zlecenie od procesora, sprawdzając bit gotowości zlecenia zapisu. Należy w tym miejscu podkreślić różnicę pomiędzy bitem gotowości urządzenia w rejestrze stanu oraz bitem gotowości zlecenia (nazywanym krótko bitem zlecenia) w rejestrze sterowania. Procesor, realizując program modułu sterującego, sprawdza, czy urządzenie jest dostępne (czy bity gotowości i zajętości są skasowane). Dla uproszczenia przyjęto, że robi to również w trybie odpytywania. Biorąc pod uwagę fakt, że procesor oczekuje na zakończenie operacji wejścia-wyjścia, odczytując rejestr stanu, nie wydaje się możliwe, aby urządzenie było zajęte, a procesor realizował kolejne zlecenie. Sytuacja taka mogłaby ewentualnie mieć miejsce w systemach wieloprocesorowych lub w przypadku odpytywania okresowego.

Operacja wyjścia polega na zapisaniu rejestru danych wyjściowych, a następnie ustawieniu bitu gotowości polecenia zapisu. Po wydaniu zlecenia procesor przechodzi w tryb odpytywania rejestru stanu w celu sprawdzenie bitu gotowości, a sterownik realizuje zlecenie. Po zrealizowaniu zlecenia sterownik ustawia bit gotowości (na co czeka procesor) oraz kasuje bit zajętości i przechodzi do oczekiwania na kolejne zlecenie.



Interakcja w przypadku operacji wejścia wygląda podobnie z tą różnicą, że procesor przed zleceniem operacji nie zapisuje rejestru danych, tym samym sterownik go nie odczytuje. Po wykonaniu operacji sterownik zapisuje rejestr danych wejściowych, który odczytuje procesor po stwierdzeniu gotowości urządzenia.



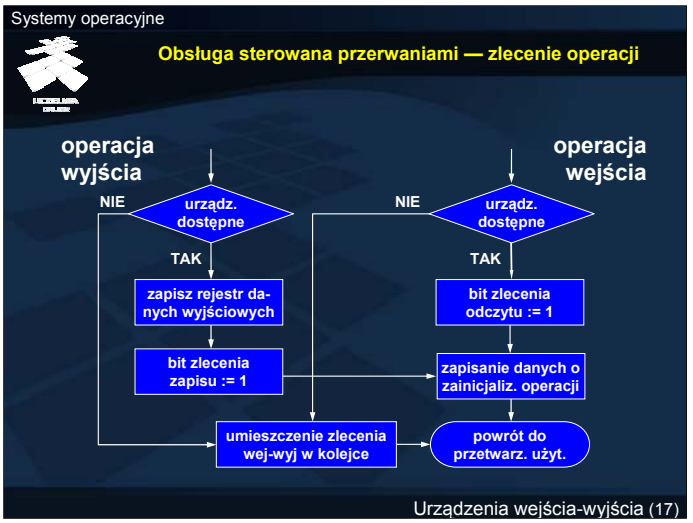
Program obsługi urządzenia, wykonywany przez procesor, polegający na testowaniu i ustawianiu odpowiednich bitów, zapisie i odczycie rejestrów danych, ewentualnie interpretacji kodów błędu jest zawarty w module sterującym. Jak wynika z wcześniejszej analizy, dopóki urządzenie nie zakończy pracy, procesor wykonuje program modułu, polegający na testowaniu bitu gotowości (ewentualnie bitów błędu). Z punktu widzenia przetwarzania aplikacyjnego jest to czas marnowany na oczekiwaniu. Pomimo, że sterownik może funkcjonować równolegle z procesorem, tylko jedno z tych urządzeń działa w danej chwili **efektywnie**.

W przypadku sterowania przerwaniami operacja wejścia-wyjścia przebiega w dwóch fazach:

- zlecenie operacji wejścia-wyjścia,
- reakcji na gotowość urządzenia w ramach procedury obsługi przerwania.

W międzyczasie centralny procesor może realizować przetwarzanie użytkowe. Proces, który zlecił wykonanie operacji zostaje najczęściej zablokowany w oczekiwaniu na zakończenie, ale dostępny może być inny proces gotowy. W ten sposób można równoważyć obciążenie systemu, co było głównym celem wprowadzenie wielozadaniowości i związanej z nią koncepcji procesu. Nawet gdyby nie było żadnego procesu gotowego, to procesor może wykonać pewne zadania systemowe, np. zapisać profilaktycznie na urządzeniu wymiany zawartość brudnych ramek pamięci fizycznej. W ostateczności będzie wykonywana pętla bezczynności, zwana również wątkiem bezczynności. Pętla bezczynności jest oczekiwaniem na przerwanie, a więc specyficznym odpytywaniem linii wejściowej przerwań w procesorze.

Po zakończeniu pracy urządzenia wejścia-wyjścia sterownik, oprócz ustawienia odpowiednich bitów stanu, zgłasza przerwanie. Pośrednikiem w przekazywaniu przerwania do procesora jest sterownik przerwań. Ma on kilka linii wejściowych i jedną wyjściową, za pośrednictwem której przekazuje sygnał na odpowiednie wejście procesora. Obsługa urządzenia po zakończeniu pracy realizowana w reakcji na przerwanie. Zanim nastąpi obsługa konieczne jest zidentyfikowanie źródła przerwania. Nie zawsze też całość obsługi urządzenia realizowana jest w procedurze obsługi przerwania. Niektóre czynności mogą zostać odroczone i wykonane poza procedurą obsługi przerwania.



Zlecenie sterownikowi operacji wejścia-wyjścia przebiega podobnie, jak w przypadku odpytywania. Tutaj jednak możliwy jest przypadek, że urządzenie nie jest dostępne dla procesu, gdyż wykonuje operację, zleconą przez inny proces. W takiej sytuacji musi nastąpić umieszczenie zlecenia lub zlecającego procesu w kolejce. Można również przyjąć ogólne podejście z kolejkowaniem operacji, niezależnie od stanu sterownika. Zadaniem modułu sterującego jest po prostu pobrać kolejne zadanie z kolejki.

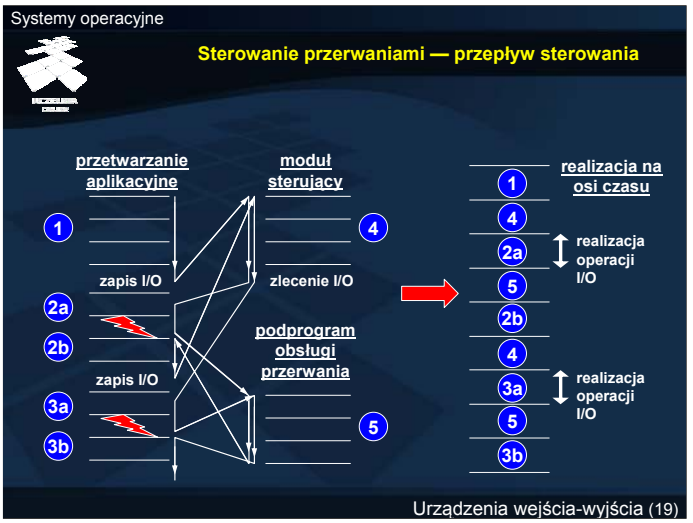
Po zleceniu operacji następuje zapisanie informacji o tej operacji w tablicy urządzeń i powrót do przetwarzania aplikacyjnego. Jest to jedna część realizacji operacji wejścia-wyjścia. Implementacja tej części określana jest w module sterującym jako górna połowa. (Nieco inaczej termin górna i dolna połowa rozumiany jest w systemie Linux).

Oprogramowanie odpowiedzialne za obsługę przerwania może być różnie zorganizowane w zależności od rozwiązań na poziomie architektury komputera. Najczęściej wyodrębnia się podprogram obsługi przerwania (ang. interrupt handler) oraz wywoływany przez niego podprogram obsługi urządzenia (ang. device handler).

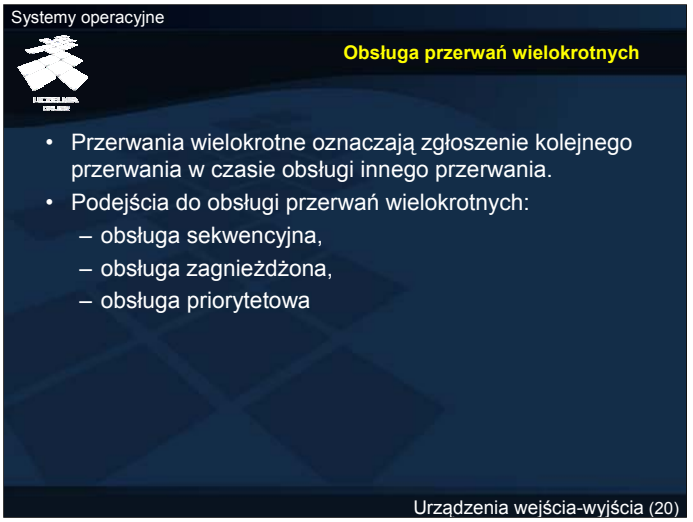
Podprogram obsługi przerwania wykonuje pewne zadania ogólne, niezależne od źródła przerwania oraz identyfikuje źródło przerwania. Właściwą obsługę urządzenia realizuje podprogram obsługi urządzenia, określany jako dolna połowa modułu sterującego. Wykonanie dolnej połowy polega na odczytaniu odpowiednich informacji z tablicy urządzeń, a następnie sprawdzeniu stanu sterownika (stanu zakończenia operacji) i przekazaniu danych i/lub informacji statusowych do procesu zlecającego wykonanie operacji.

Jeśli w kolejce do urządzenia znajdują się kolejne żądania, wybierane jest jedno z nich i zlecana jest następna operacja wejścia-wyjścia.

Procedura obsługi przerwania musi być wykonana dość szybko ze względu na blokowanie obsługi innych przerwań. Pewne czasochłonne zadania, związane z przetwarzaniem danych w ramach operacji wejścia-wyjścia, mogą być wykonane później, poza obsługą przerwania. Określa się je jako czynności odroczone, a w systemie Linux nazywa dolną połową, podczas gdy górna połowa oznacza część kodu, wykonywaną bezpośrednio w reakcji na przerwanie. Przykładem czynności odroczonych jest interpretacja zawartości ramki, odebranej przez kartę sieciową. Zawartość bufora karty musi być skopiowana możliwie szybko, a dalsze przetwarzanie, np. sprawdzenie poprawności, interpretacja adresów itp., mogą być wykonane nieco później.



W przedstawionym schemacie przepływu sterowania górna połowa modułu kończy się po wydaniu zlecenia dla urządzenia wejścia-wyjścia, po czym jest powrót do przetwarzania aplikacyjnego. Gotowość urządzenia sygnalizowana jest przez przerwanie, oznaczone „błyskawicą” i sterowanie przechodzi do dolnej połowy modułu sterującego. Po obsłużeniu przerwania kontynuowane jest przetwarzania aplikacyjne. W tym przypadku możliwe jest jednoczesne przetwarzanie użytkowe i praca urządzenia wejścia-wyjścia.

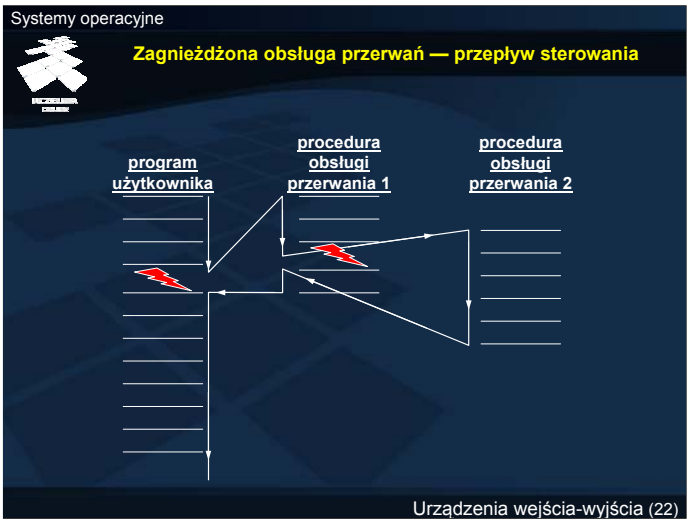


Przerwania wielokrotne są zjawiskiem naturalnym w przypadku jednoczesnej obsługi wielu urządzeń. Przykładem wystąpienia przerwań wielokrotnych jest odbiór danych z łącza komunikacyjnego w czasie obsługi drukarki.

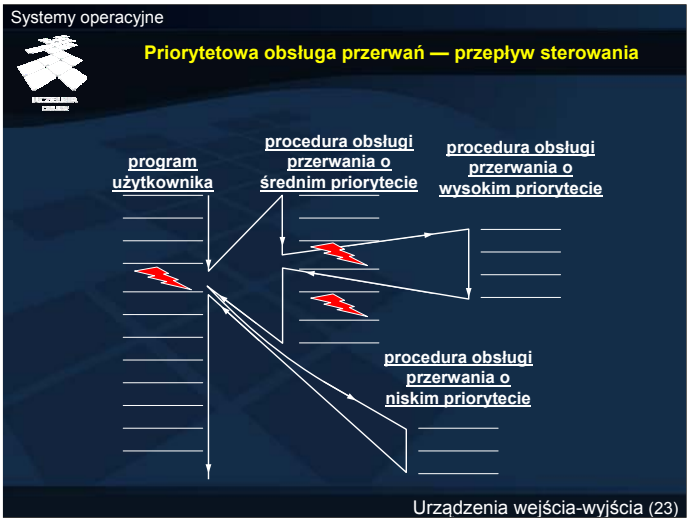
Ogólnie można wyróżnić kilka podejść do obsługi przerwań wielokrotnych:

- obsługa sekwencyjna — kolejne przerwanie (zgłoszone podczas obsługi) obsługiwane jest po zakończeniu obsługi bieżącego,
- obsługa zagnieżdżona — po zgłoszeniu nowego przerwania obsługa bieżącego jest zawieszana i kontynuowana po obsłużeniu przerwania nowo zgłoszonego,
- obsługa priorytetowa — zawieszenie obsługi bieżącego przerwania następuje tylko wówczas, gdy nowo zgłoszone przerwanie ma wyższy priorytet, w przeciwnym razie obsługa następuje po obsłużeniu wszystkich zgłoszonych przerwań o wyższym priorytecie

W czasie obsługi przerwania 1 zgłaszane jest kolejne przerwanie, ale reakcja na nie następuje dopiero po zakończeniu procedury obsługi przerwania 1. Zaznaczono to jako powrót do przetwarzania aplikacyjnego, ale zanim nastąpi wykonanie kolejnej instrukcji programu użytkownika, wykonana zostanie procedury obsługi przerwania 2.

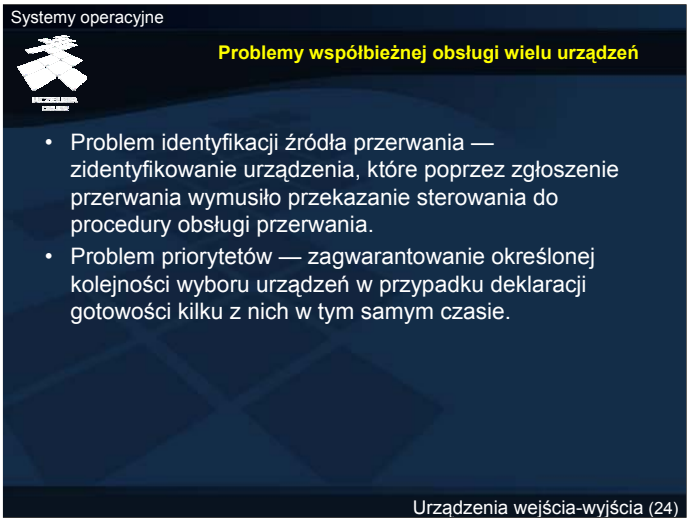


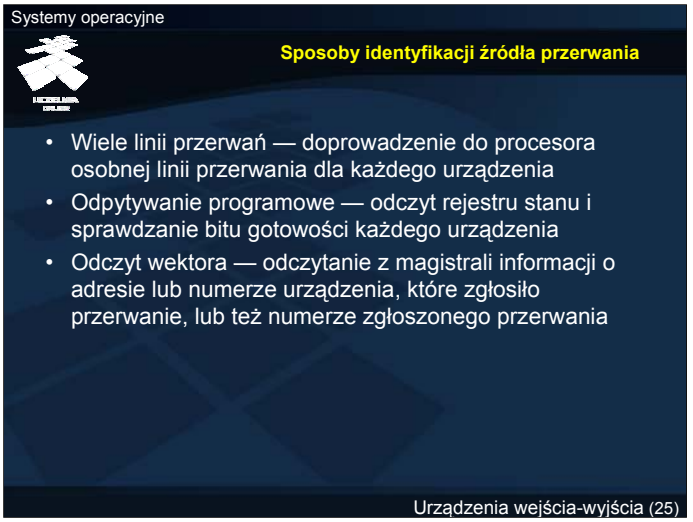
Zgłoszenie kolejnego przerwania podczas obsługi przerwania 1 powoduje przerwanie bieżącej procedury i przejście do obsługi przerwania 2. Po zakończeniu procedury obsługi przerwania 2 kontynuowana jest obsługa przerwania 1, a po jej zakończeniu sterowanie wraca do programu użytkownika.



W czasie obsługi przerwania o średnim priorytecie następuje zgłoszenie przerwania o wysokim priorytecie, a następnie przerwania o niskim priorytecie. W pierwszym przypadku postępowanie jest zgodne ze schematem dla obsługi zagnieżdżonej, a w drugim dla obsługi sekwencyjnej.

Podejście priorytetowe jest oczywiście powszechnie stosowane w obsłudze przerwań.


Systemy operacyjne
Problemy współbieżnej obsługi wielu urządzeń
• Problem identyfikacji źródła przerwania — zidentyfikowanie urządzenia, które poprzez zgłoszenie przerwania wymusiło przekazanie sterowania do procedury obsługi przerwania.
• Problem priorytetów — zagwarantowanie określonej kolejności wyboru urządzeń w przypadku deklaracji gotowości kilku z nich w tym samym czasie.
Urządzenia wejścia-wyjścia (24)

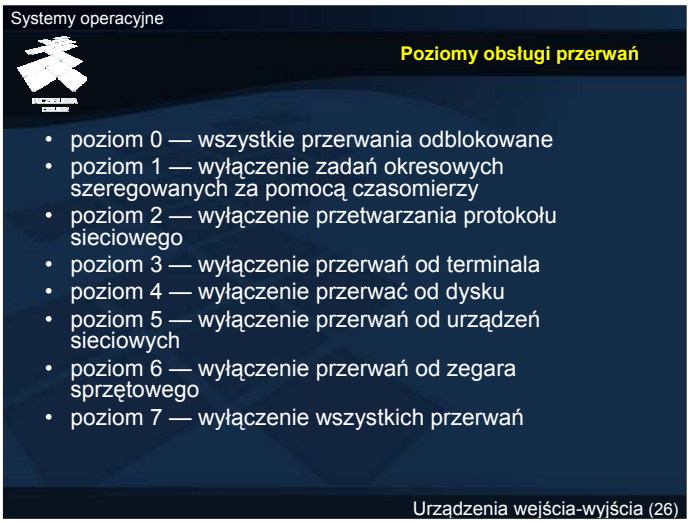


Podejście z wieloma liniami przerwań wymagałoby tylu linii przerwań, ile jest urządzeń mogących zgłosić przerwanie. Linii tych może być niemal zawsze albo za dużo albo za mało, podejście jest więc trudne do zrealizowania w systemie ogólnego zastosowania.

Odpytywanie programowe wymaga odczytania i zinterpretowania stanu każdego urządzenia. Można się oczywiście ograniczyć do tych urządzeń, dla których w tablicy urządzeń jest odnotowany fakt realizacji operacji. Wymaga to jednak przeglądania tablicy urządzeń, ogólnie jest więc czasochłonne.

Odczyt wektora wymaga umieszczenia odpowiednich informacji (numeru urządzenia, adresu urządzenia lub numeru przerwania) na magistrali. Informacja może zostać wystawiona po potwierdzeniu otrzymania przerwania przez procesor (odpytywanie sprzętowe). Sygnał potwierdzający otrzymanie przerwania propagowany jest łańcuchowo przez urządzenia aż do napotkania tego, które zgłosiło przerwanie. Innym sposobem jest uzyskanie wyłączności dostępu do magistrali i wystawienie odpowiedniego wektora przed zgłoszeniem przerwania (arbitraż na magistrali).

Rozwiązaniem hybrydowym jest użycia sterownika przerwań. Ma on wiele linii wejściowych i jedną wyjściową, podłączoną do odpowiedniego wejścia procesora. Po stwierdzeniu przerwania wystarczy odpytać sterownik przerwań, na której linii nastąpiło zgłoszenie. Sterownik przerwań ma jednak taką samą wadę, jak procesor, jeśli chodzi o liczbę wejściowych linii przerwań. Dlatego podejście to łączy się z odpytywaniem programowym. Wiele urządzeń może zgłaszać przerwanie o tym samym numerze. Obsługa przerwania rozpoczyna się od zidentyfikowania numeru przerwania, a następnie odpytywane są wszystkie urządzenie, które zgłaszają przerwanie o tym numerze. W praktyce wygląda to tak, że z każdym numerem przerwanie związany jest łańcuch modułów sterujących. W ramach obsługi przerwania uruchamiana jest odpowiednia procedura kolejnego modułu, a jaj zadaniem jest stwierdzenie, czy urządzenie przez nią obsługiwane uzyskało stan gotowości.

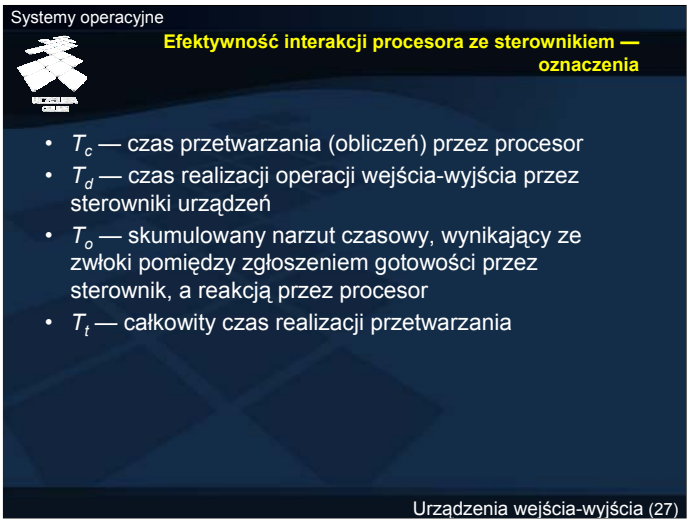


Priorytety urządzeń wiążą się z priorytetami przerwań od nich, ale problem można uogólnić na priorytety innych zdarzeń, na które jądro musi reagować. W związku z powyższym wyróżnia się poziomy pracy jądra związane z reakcją na pewne zdarzenia. Na określonym poziomie pracy (poziomie priorytetu) jądro reaguje tylko na zdarzenia o wyższym priorytecie. Obsługa przerwania rozpoczyna się od odpowiedniego podniesienia poziomu pracy. W zaprezentowanym przykładzie (zbliżonym do rozwiązań w systemach rodziny UNIX):

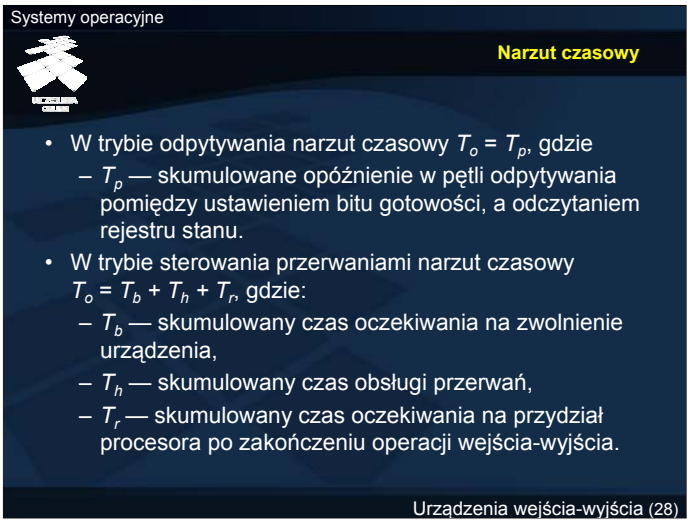
- na poziomie 0 jądro reaguje na wszystkie zdarzenia (przerwania),
- na poziomie 1 ignoruje (kolejkuje, maskuje) żądania realizacji zadań okresowych,
- na poziomie 2 nie przetwarza danych protokołu sieciowego, ale obsługuje kartę sieciową (zaprzestaje reakcji dopiero na poziomie 5)
- na poziomie 3 nie obsługuje żądań od terminala,
- na poziomie 4 nie obsługuje żądań od dysku,
- na poziomie 5 nie obsługuje żądań od karty sieciowej,
- na poziomie 6 nie reaguje na przerwania od czasomierza (generatora interwałów),
- na poziomie 7 nie reaguje na żadne przerwania.

Na każdym wyższym poziomie jądro oczywiście nie reaguje również na zdarzenia przypisane do niższego poziomu.

W konkretnych implementacja tych poziomów może być więcej. W systemie Windows 2000/XP występują 32 tzw. *poziomy zgłoszeń przerwań*. W systemach Linux i Solaris do obsługi przerwań wykorzystywane są wątki jądra i ich priorytet decyduje o realizacji określonej procedury.



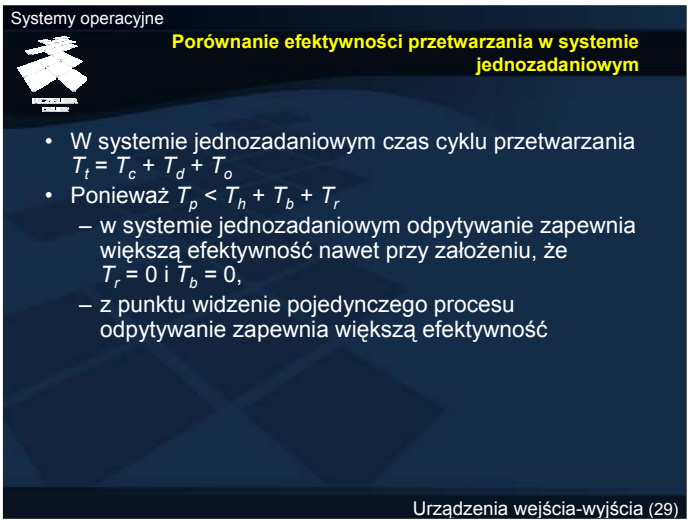
W celu porównania efektywności przetwarzania w systemie z odpytywaniem oraz z przerwaniami należy wrócić do podstawowego cyklu zmian stanów procesu. Stan procesu zmienia się cyklicznie z *wykonywanego* na *oczekujący*, następnie *gotowy* i znowu *wykonywany*. Na poziomie architektury procesor oraz urządzenia wejścia-wyjścia są albo bezczynne, albo zajęte obsługą. Niech T_c oznacza zatem całkowity czas przetwarzania przez procesor, a T_d całkowity czas realizacji operacji wejścia-wyjścia. Czasy te są niezależne od sposobu obsługi urządzeń. Składnikiem, zależnym od sposobu obsługi urządzeń jest narzut czasowy — T_o. Narzut ten wynika ze zwłoki w reakcji procesora na zgłoszenie gotowości urządzenia lub z oczekiwania na dostępność urządzenia.



W odpytywaniu występuje opóźnienie, wynikające z różnicy czasu pomiędzy ustawieniem bitu gotowości, a odczytaniem rejestru stanu sterownika przez procesor. Narzut czasowy w przypadku odpytywania wynika wyłącznie z tego opóźnienia.

W przypadku przerwań narzut wynika z:

- czasu oczekiwania na zwolnienie urządzenia (urządzenie może być zajęte przez inny, oczekujący proces),
- czasu obsługi przerwania (potwierdzanie, identyfikacja źródła),
- czasu oczekiwania na przydział procesora, po zakończeniu operacji wejścia-wyjścia (po uzyskaniu gotowości).

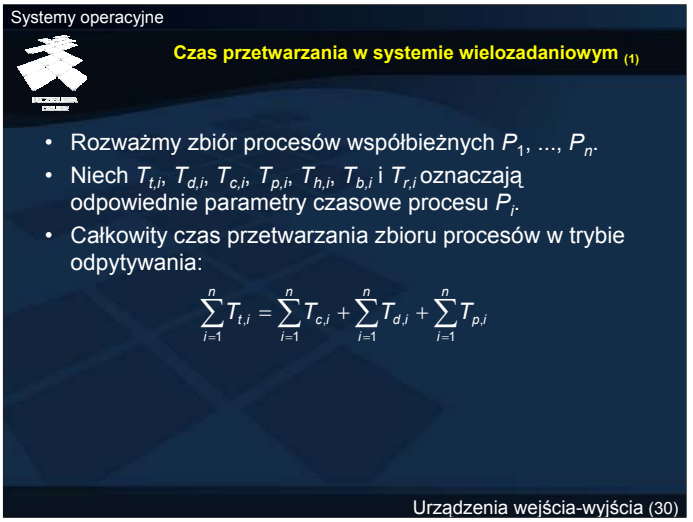


W systemie jednozadaniowym czas cyklu przetwarzania — T_t — jest sumą:

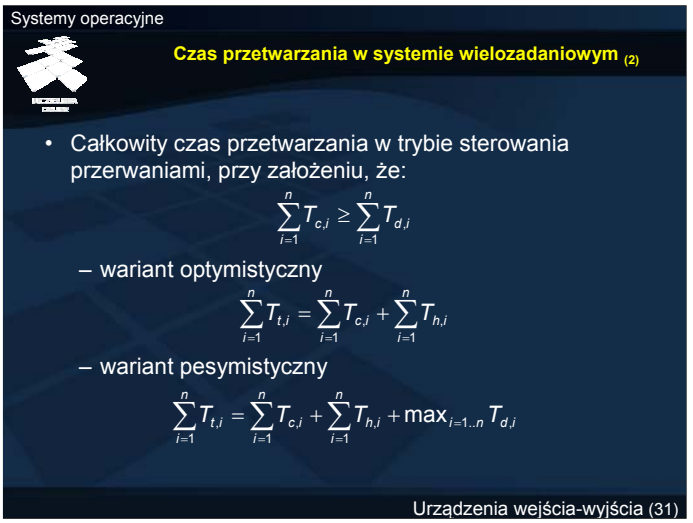
- czasu obsługi przez procesor — T_c,
- czasu oczekiwania na zakończenie operacji wejścia-wyjścia (gotowość urządzeń) — T_d,
- narzutu czasowego — T_o.

Różnica pomiędzy odpytywaniem a sterowaniem przerwaniami jest tylko w narzucie czasowym, który jest mniejszy w przypadku odpytywania.

Rozważając system wielozadaniowy, jeśli proces uzyska procesor, odpytywanie jest dla niego korzystniejsze, gdyż narzut czasowy jest mniejszy, a oczekiwanie na gotowość urządzenia jest z **jego** punktu widzenia czasem marnowanym (nie ma wówczas postępu w realizacji programu).



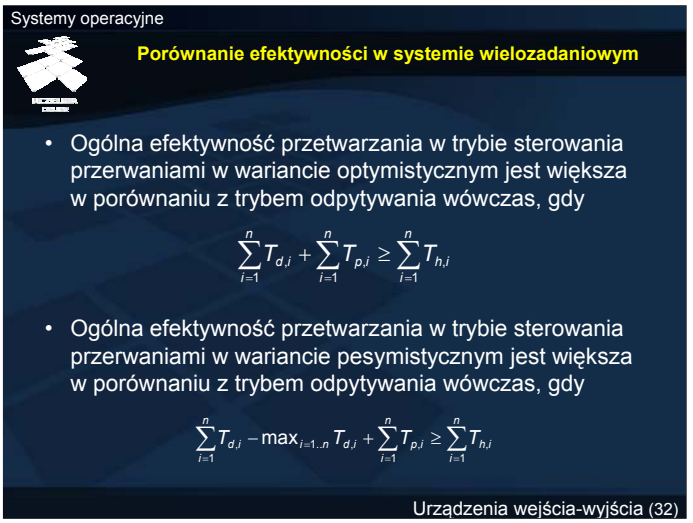
Na potrzeby systemu wielozadaniowego analizowany jest łączny czas przetwarzania zbioru procesów. W przypadku odpytywania nie ma jednoczesności przetwarzania, a więc czas ten jest sumą czasów przetwarzania poszczególnych procesów.



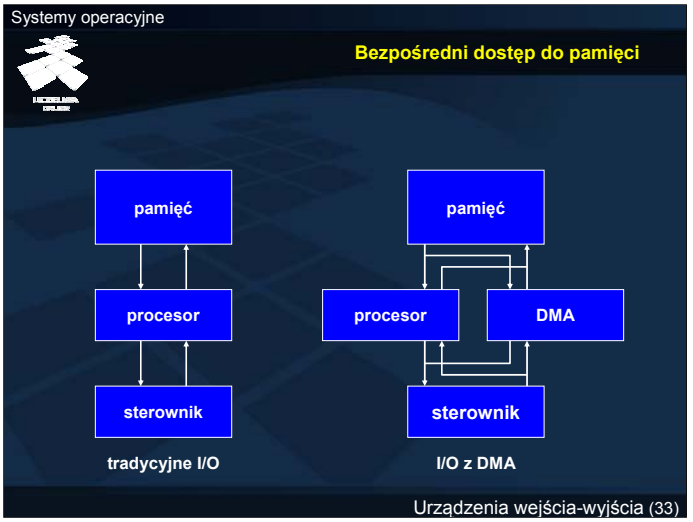
Analiza w trybie sterowania przerwaniami zrobiona jest przy założeniu, że łączny czas procesora, potrzebny do realizacji przetwarzania jest większy, niż łączny czas realizacji operacji wejścia-wyjścia. Zbiór zadań jako całość ograniczony jest więc procesorem.

W wariancie optymistycznym zawsze po wejściu w stan oczekiwania procesu zlecającego operację wejścia-wyjścia jest jakiś inny proces gotowy. Oznacza to, że w czasie, gdy proces oczekuje na dostępność urządzenia lub przydział procesora, procesor wykonuje inne zadanie. Jedynie sama obsługa przerwania angażuje procesor i w tym czasie nie może on wykonywać przetwarzania aplikacyjnego.

W wariancie pesymistycznym wszystkie procesy w tym samym czasie zlecają wykonanie operacji wejścia-wyjścia na tym samym urządzeniu, a urządzenie przydzielane jest temu procesowi, którego zlecenie jest najbardziej czasochłonne.



Czas skumulowanej obsługi przerwań, nawet w wariancie pesymistycznym, musiałby być znaczący, żeby kwestionować ogólną zasadność obsługi urządzeń wejścia-wyjścia sterowanej przerwaniami. Takie podejście mogłoby jedynie mieć sens w przypadku bardzo szybkich urządzeń, które potrafią zrealizować operację wejścia-wyjścia w czasie kilkunastu lub kilkudziesięciu cykli rozkazowych.



Układ DMA ma „kompetencje" procesora w zakresie dostępu do pamięci, w związku z czym może rywalizować z procesorem o dostęp do magistrali systemowej w celu przejęcia sterowania systemem komputerowym.

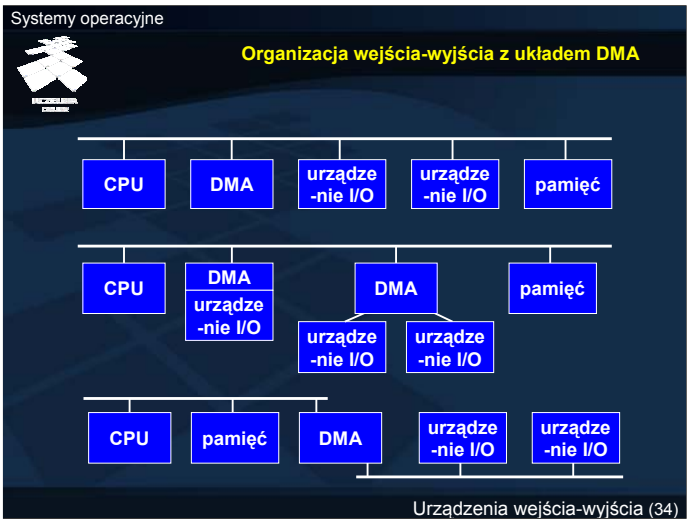
Układ DMA wykorzystywany jest w celu realizacji transferu większych bloków danych, np. w przypadku dysku lub karty sieciowej.

Procesor zleca układowi DMA transmisję danych, przekazując następujące parametry:

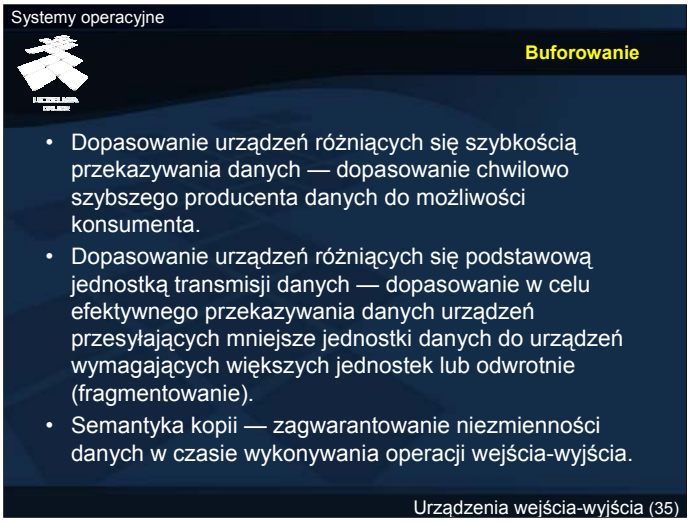
- rodzaj operacji (zapis lub odczyt bloku w pamięci),
- adres urządzenia wejścia-wyjścia,
- początkowy adres bloku w pamięci do zapisu/odczytu,
- rozmiar zapisywanego/odczytywanego bloku w bajtach lub słowach.

W celu realizacji zlecenia układ DMA przejmuje kontrolę nad magistralą, gdy nie jest ona potrzebna procesorowi lub „wykrada" cykl magistrali procesorowi i realizuje przesłanie słowa. Fakt zakończenia transmisji bloku danych układ DMA sygnalizuje procesorowi, **zgłaszając przerwanie**.

Pomimo ograniczeń w jednoczesnej pracy procesora i układu DMA, wynikającej z konieczności zapewnienia wyłącznego dostępu do magistrali, realizacja transferu bloku z udziałem DMA daje poprawę efektywności. Przekazanie bloku słowo po słowie przez procesor, przeplatane z realizacją przetwarzania aplikacyjnego, wymaga przerwania, powrotu z przerwania, zmiany zawartości niektórych rejestrów (przechowania i odtworzenia odpowiednich wartości). Poza tym procesor korzysta z pamięci podręcznej i nie zawsze wymagany jest dostęp do magistrali systemowej w cyklu rozkazowym.



Układ DMA może być różnie umiejscowiony w architekturze komputera. W pierwszej z przedstawionych konfiguracji wszystkie podstawowe układy, czyli procesor, pamięć, urządzenia wejścia-wyjścia oraz DMA współdzielą magistralę systemową. Układ DMA musi wówczas rywalizować o magistralę z procesorem zarówno przy dostępie do pamięci, jak i przy dostępie do urządzeń wejścia-wyjścia. W pozostałych konfiguracjach rywalizacja z procesorem występuje tylko przy dostępie do pamięci. W drugiej konfiguracji DMA jest ściśle zintegrowany z urządzeniem, które obsługuje. Może też obsługiwać kilka takich urządzeń. W trzeciej konfiguracji wyodrębniono magistralę wejścia-wyjścia, dostępną za pośrednictwem układu DMA.



Ogólnym celem buforowania jest niwelowanie różnic, co oddaje angielskie znaczenie tego terminu (ang. buffer = zderzak). Niwelowanie różnic polega na wykorzystaniu szybkiej pamięci operacyjnej do gromadzenia danych, zanim zostaną one przekazane do urządzenia.

Różnice mogą dotyczyć szybkości pracy urządzeń. Typowy przypadek dotyczy szybkiego producenta, przekazującego dane, z których przetwarzaniem nie nadąża odbiorca — konsument. Czasami, w celu równoważenia obciążenia urządzenia szybsze otrzymują więcej pracy do wykonania, w związku z czym nie mogę natychmiast reagować na żądania urządzeń powolnych. Wówczas również przydaje się buforowanie.

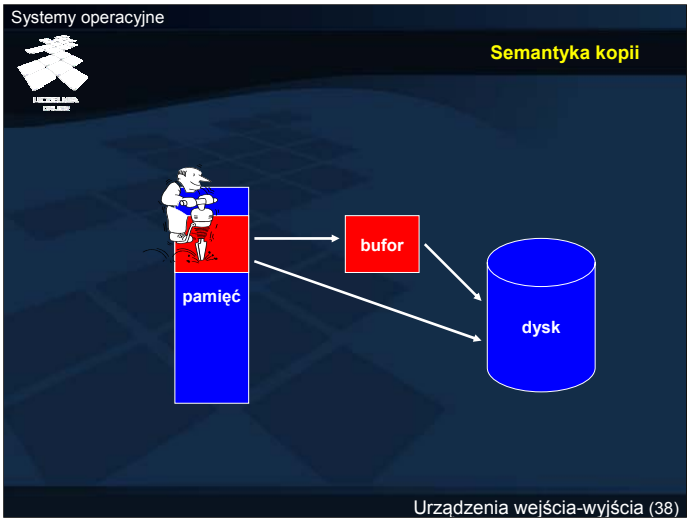
Inny przypadek dotyczy efektywności transmisji danych pomiędzy urządzeniami blokowymi. Bloki przekazywane przez jedno urządzenie mogą mieć innych rozmiar, niż bloki akceptowane przez inne urządzenie. W celu uniknięcia zbędnych operacji wejścia-wyjścia dane gromadzi się w buforze w celu „uformowania” jednostki o właściwym rozmiarze.

Nieco odmiennym przypadkiem jest semantyka kopii. Przypadek ten wiąże się z synchronizacją współbieżnego dostępu, a dokładniej koniecznością uniknięcia długotrwałego blokowania dostępu do danych. W tym celu robi się kopie danych w pamięci, które następnie można przekazać do urządzenia, podczas gdy proces może operować na danych oryginalnych.

Przykładem dopasowania szybkiego producenta do stosunkowo małej przepustowości konsumenta jest drukowanie. Drukarka jest urządzeniem powolnym, podczas gdy komputer potrafi przygotować dane do wydruku dość szybko, nawet w przypadku dokumentu ze skomplikowanym formatowaniem, rysunkami itp. Strumień, który ma być przekazany do drukarki jest więc gromadzony w pamięci, co umożliwia po stosunkowo krótkim czasie kontynuacje pracy z drukowanym właśnie dokumentem.

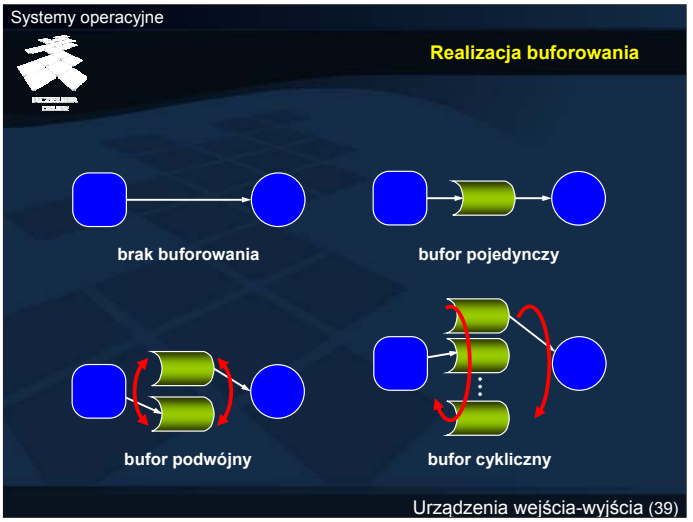
W obsłudze drukarki stosowany jest dodatkowo tzw. spooler, co zostanie omówione z dalszej części.

Przykładem dopasowania jednostek transmisji jest zapis na dysku danych odbieranych z sieci. Po wyodrębnieniu z ramki sieciowej danych aplikacyjnych może się okazać, że ich objętość jest mniejsza niż rozmiar sektora. Dane można oczywiście zapisać, tzn. zapisać sektor dysku, przy czym część sektora będzie zwierać wartości przypadkowe lub 0. Po otrzymaniu kolejnej ramki dane w sektorze można uzupełnić, ale operacja taka wymaga wczytania sektora do pamięci — do bufora, uzupełnienia świeżo otrzymanymi danymi i ponownego zapisu. Buforowanie zatem i tak jest niezbędne na potrzeby realizacji operacji dyskowych. Tym nie mniej, lepiej przetrzymać niepełną zawartość bufora w pamięci i poczekać na uzupełnienie danymi z następnej ramki. W tym przypadku mamy do czynienia ze składaniem. Przy transmisji danych z dysku przez sieć w opisanej sytuacji może być z kolei potrzeba fragmentowania danych, czyli dzielenia w buforze na mniejsze części, dopasowane do pojemności ramki.

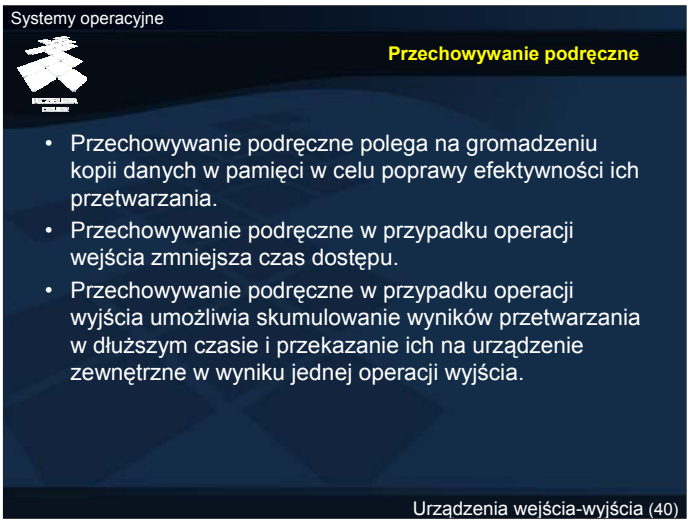


Przykładem buforowania w celu zagwarantowania semantyki kopii jest zapis na dysku pliku, który podlega modyfikacji. Przypadek taki ma często miejsce, kiedy chcemy zachować bieżący stan dokumentu na wypadek awarii lub przed jakąś poważną modyfikacją, której może się nie udać wycofać. Transfer danych bezpośrednio z przestrzeni procesu oznacza ryzyko zapisania danych niespójnych. Ryzyko niespójności w tym przypadku polega na tym, że w trakcie realizacji operacji zapisu (wyjścia) plik w pamięci jest modyfikowany i część zapisanych zmian dotyczy zawartości sprzed modyfikacji, a część po modyfikacji. Można w ten sposób zapisać na dysku drugą połowę wprowadzanego właśnie zdania, pomimo że zlecenie zapisu zostało wydane przed rozpoczęciem wprowadzania tego zdania.

Można by oczywiście zablokować dostęp do obszaru pamięci z bieżącą zawartością pliku, ale zmusza to użytkownika do przerwania pracy na czas zapisu na dysku. Można zatem wykonać znacznie szybszą operację skopiowania odpowiedniego obszaru pamięci do bufora w jądrze, gdzie będzie gwarancja niezmienności danych.



Buforowanie można zrealizować na kilka sposobów, w zależności od zastosowania i intensywności korzystania z buforów. Całkowita rezygnacja z buforowania oznacza konieczność utrzymania urządzeń wyjściowych w gotowości do natychmiastowego działania lub blokowania pracy urządzeń wejściowych. Można zastosować bufor pojedynczy, który ma jednak pewne ograniczenie — albo jest w danej chwili czytany, albo zapisywany. W celu umożliwienia jednoczesnej pracy obu komunikującym się urządzeniom konieczne jest użycie co najmniej bufor podwójnego. Jedna z dwóch pozycji bufora jest w danej chwili zapisywana, a druga odczytywana, po czym następuje zmiana. W przypadku dużych tymczasowych rozbieżności w szybkości pracy urządzeń konieczny może być bufor cykliczny, w którym jest kilka pozycji. Producent zapełnia kolejne pozycje, a gdy dojdzie do końca, rozpoczyna ponownie od początku. Podobnie konsument pobiera dane z poszczególnych pozycji. Warunkiem poprawności takiego buforowania jest blokada producenta przed nadpisaniem pozycji, której konsument jeszcze nie odczytał oraz blokowanie konsumenta przed odczytaniem pozycji, której producent jeszcze nie zapisał (wielokrotnym odczytaniem tej samej pozycji). Tego typu problem rozważany będzie w module dotyczącym synchronizacji procesów.



Przechowywanie podręczne jest pewną formą buforowania (raczej utrzymywania danych w pamięci), której celem jest poprawa efektywności realizacji operacji wejścia-wyjścia poprzez udostępnianie danych w pamięci zamiast wykonywania operacji na urządzeniu.

W przypadku zlecenia odczytu danych z urządzenia są one przekazywane procesowi z bufora w pamięci podręcznej, z pominięciem operacji wejścia. Wymaga to utrzymania aktualnych danych w buforze pamięci podręcznej, a więc w wyniku operacji zapisu na urządzeniu musi nastąpić stosowana zmiana zawartości tego bufora. Operacja wyjścia może się ograniczyć do modyfikacji zawartości bufora. Właściwa operacja wyjścia na urządzeniu może nastąpić później i uwzględnić kilka zleceń zapisu ze strony procesów aplikacyjnych.

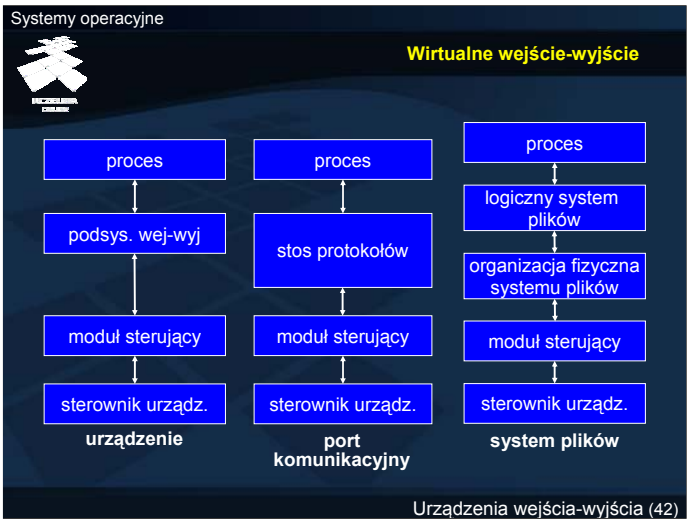
Tego typu technika stosowana jest powszechnie w obsłudze systemu plików, co będzie omawiane w następnych modułach. Jest ona również stosowana w obsłudze pamięci wirtualnej, ale w tym przypadku odnosi się do **zawartość pamięci**, urządzenie wymiany jest zatem wtórnym miejscem gromadzenia danych.

Spooling jest akronimem od angielskiego określenia *sequential peripheral operation on line*, co można przetłumaczyć, oddając sens sformułowania, jako *na bieżąco realizowana operacja sekwencyjna na urządzeniu wejścia-wyjścia*.

Operacja sekwencyjna jest tutaj rozumiana jako operacja na urządzeniu o dostępie wyłącznym. Następna operacja może się rozpocząć dopiero po zakończeniu poprzedniej. Konieczne jest więc zagwarantowanie ciągłości strumienia danych, przekazywanych do urządzenia w ramach operacji wyjścia. Strumień taki nie może się przeplatać z innym strumieniem. Typowym przykładem urządzenia, obsługiwanego z wykorzystaniem techniki spoolingu, jest drukarka. Przeplatanie strumieni w przypadku takiego urządzenia oznacza wydruk, który może być zupełnie nieczytelny (pomieszanie linii, słów, znaków lub zupełna abstrakcja w przypadku grafiki).

Spooling polega więc na buforowaniu strumienia danych, przekazywanych do urządzenia dostępnego w trybie wyłącznym i rozpoczęciu operacji dopiero po zapamiętaniu całego strumienia. Ze względu na objętość strumienia buforowanie takie odbywa się najczęściej na dysku (tzw. spooler). Przekazywanie danych ze spoolera do urządzenia realizuje jeden proces lub wątek, co zapewnia sekwencyjność tej operacji.

Alternatywą dla spoolingu byłoby blokowanie urządzenia, tzn. jawny przydział urządzenia żądającemu procesowi i blokowanie dostępu innym procesom do czasu zwalniania. Takie rozwiązanie stwarza jednak problem uzależnienie zajętości urządzenia od „kaprysu” procesu. Jeśli proces przekaże tylko część strumienia, a drukowanie reszty będzie musiał odłożyć w czasie, drukarka pozostanie cały czas zajęta. W przypadku spoolingu drukowanie nie rozpocznie się, dopóki cały strumień nie zostanie zbuforowany, dzięki czemu przedłużające się przekazywanie danych nie blokuje innych procesów w dostępie do drukarki.



Chociaż podsystem wejścia-wyjścia udostępnia interfejs dla obsługi wszystkich urządzeń wejścia-wyjścia, wykorzystanie niektórych z tych urządzeń w taki „surowy” sposób jest niewygodne. Karta sieciowa na przykład jest urządzeniem, które staje się dopiero wówczas użyteczne, gdy otrzymuje dane z odpowiednimi informacjami adresowymi i kontrolnymi, stosownie do używanych protokołów sieciowych. Dysk służy do przechowywania danych w sektorach, ale logicznie dane zorganizowane są w pliki, które mogą zajmować wiele sektorów. Musi gdzieś być zatem pamiętana informacja o sektorach (lub innych jednostkach alokacji) zajmowanych przez plik.

Zarządzanie taką informacją na poziomie aplikacji naraża system na sporo błędów. Dwa różne procesy mogłyby na przykład przydzielić sobie ten sam blok dyskowy. Poza tym zadaniem jądra systemu operacyjnego jest stworzenie środowiska wygodnego dla użytkowników, ułatwiającego wykonywanie programów. Tego typu usługi są wiec realizowane przez oprogramowanie systemowe, które na bazie modułu sterującego urządzenia fizycznego tworzy pewne urządzenia wirtualne, oferujące często zupełnie inny model wykorzystania zasobów urządzeń zewnętrznych.

Dwoma nadmienionymi już przykładami są: protokoły komunikacji sieciowej i system plików. Stos protokołów udostępnia usługi transmisji danych na bazie fizycznej karty sieciowej. Dla aplikacji urządzeniem jest połączenie sieciowe a nie karta. System plików udostępnia logiczny obraz informacji w postaci hierarchicznej struktury katalogów oraz plików identyfikowanych przez nazwy, a moduł organizacji fizycznej systemu plików odwzorowuje ten obraz na zbiór sektorów fizycznego dysku. Dla aplikacji urządzeniem jest więc plik o określonej nazwie, a nie dysk.