Akademia Nauk Stosowanych
Instytut Techniczny
Kierunek: Informatyka
studia I stopnia, semestr 3

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Systemy operacyjne

WYKŁAD 5 i 6

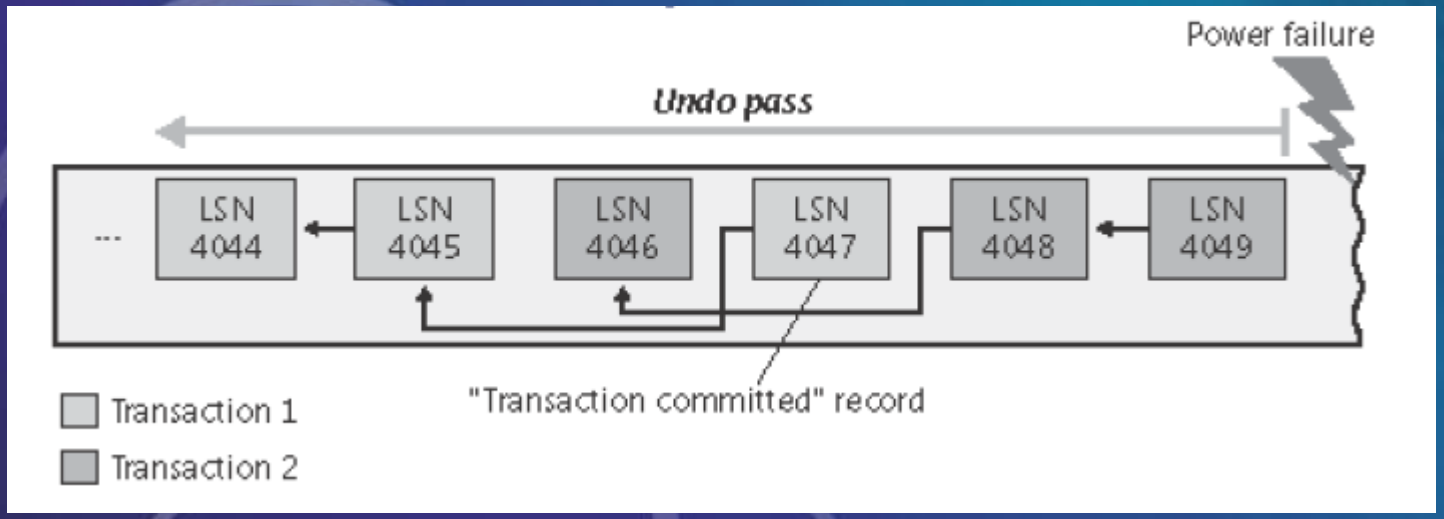
dr inż. Stanisława Plichta

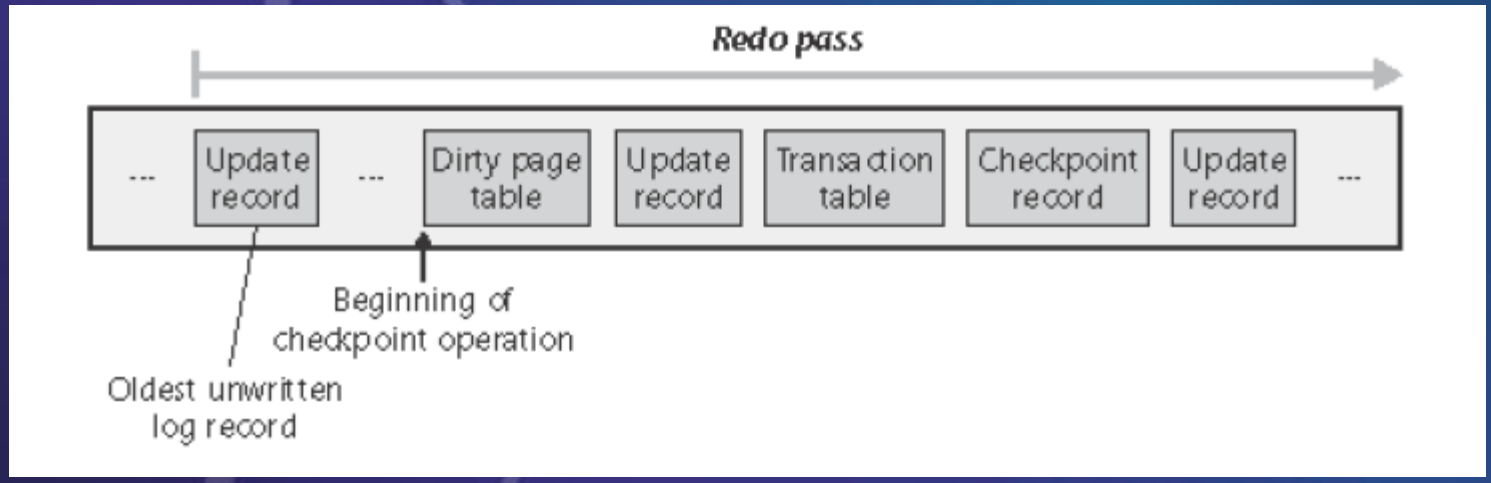
splichta@ans-ns.edu.pl

NTFS – odporność systemu

- W NTFS wprowadzono koncepcje transakcyjności operacji.
- Wszelkie operacje na danych wykonywane są w transakcjach.
- Zasadniczą techniką wspierającą odporność na awarie jest prowadzenie dziennika wszystkich operacji wykonanych na danych
- Dziennik zastosowany w NTFS jest typu undo/redo (unieważnienie/powtarzanie).

Odtwarzanie UNDO, REDO





NTFS – odporność systemu

- System plików NTFS pozwala jedynie na pełną rekonstrukcję danych systemowych, czyli: katalogów, atrybutów bezpieczeństwa, mapy bitowej zajętych klastrów oraz pozostałych plików systemowych.

Odtwarzanie polega na:

- Powtórzeniu wszystkich transakcji zatwierdzonych, zaczynając od najwcześniejszej.
- Unieważnieniu wszystkich transakcji niezakończonych, zaczynając od najpóźniejszej.

Organizacja kontroli dostępu w systemie Windows

Zezwolenia indywidualne

ZEZWOLENIE	DLA KATALOGU	DLA PLIKU
Read (R)	Odczyt nazw plików i katalogów wchodzących w skład danego katalogu, a także atrybutów oraz właściciela katalogu.	Odczyt danych, atrybutów, właściciela pliku.
Write(W)	Dodawanie plików i podkatalogów do katalogu, zmiana atrybutów katalogu, odczyt właściciela.	Odczyt właściciela, zmiana atrybutów pliku, zmiana oraz dopisanie danych do pliku.
Delete (D)	Usunięcie katalogu	Usunięcie pliku.
Change Permission (P)	Zmiana zezwoleń dla katalogu.	Zmiana zezwoleń dla pliku.
Take Ownership (O)	Zmiana właściciela katalogu.	Zmiana właściciela pliku.

autor: dr inż. Stanisława Plichta

Organizacja kontroli dostępu w systemie Windows

ZEZWOLENIA STANDARDOWE	ZEZWOLENIA INDYWIDUALNE
No Access	-
Read	R, X
Change	R, W, X, D
Full Control	R, W, X, D, P, O

Organizacja kontroli dostępu w systemie Windows

ZEZWOLENIA STANDARDOWE	ZEZWOLENIA INDYWIDUALNE DLA KATALOGÓW	ZEZWOLENIA INDYWIDUALNE DLA PLIKÓW przy DZIEDZICZENIU
No Access	-	-
List	R, X	-
Read	R, X	R, X
Add	W, X	-
Add&Read	R, W, X	R, X
Change	R, W, X, D	R, W, X, D
Full Control	wszystkie	wszystkie

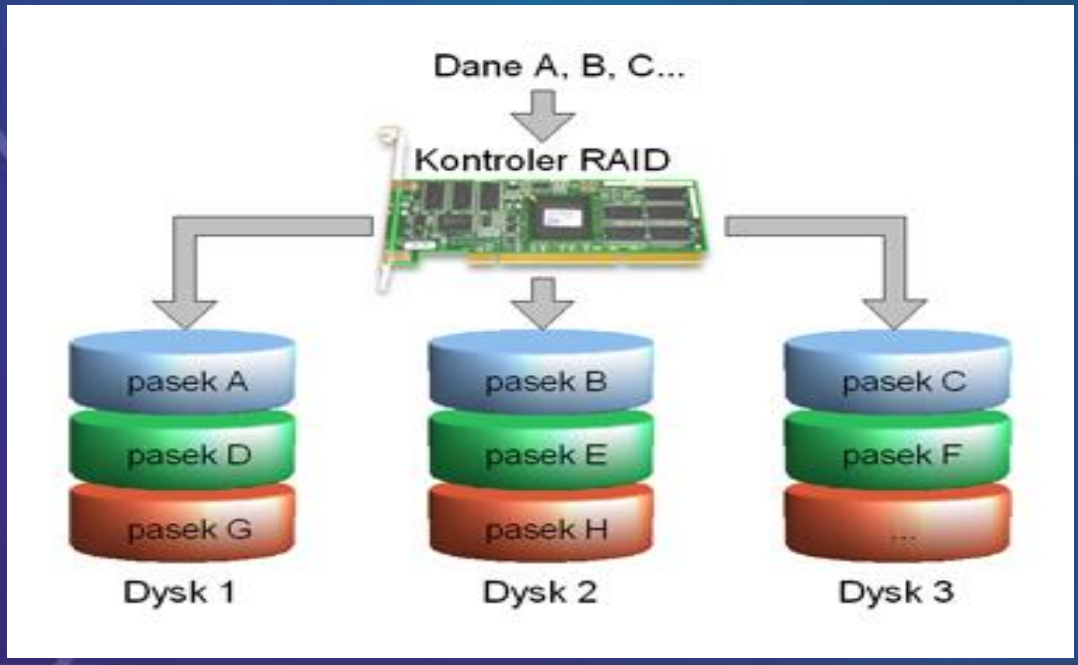
Macierze dyskowe

- Powstały w celu zabezpieczenia danych na dyskach wskutek ich awarii jak również, aby zwiększyć transfer między serwerami a dyskami.
- Pracę nad projektem macierzy zainicjowano na Uniwersytecie w Berkeley w 1987 roku. W wyniku tych prac powstała specyfikacja architektury RAID - sześć poziomów określanych RAID od 0 do 5. W późniejszym okresie dodano poziom pośredni RAID 0+1.
- RAID (Redundant Array of Independent Disks) – zbiór urządzeń dyskowych widzianych przez system jako jedno urządzenie logiczne.

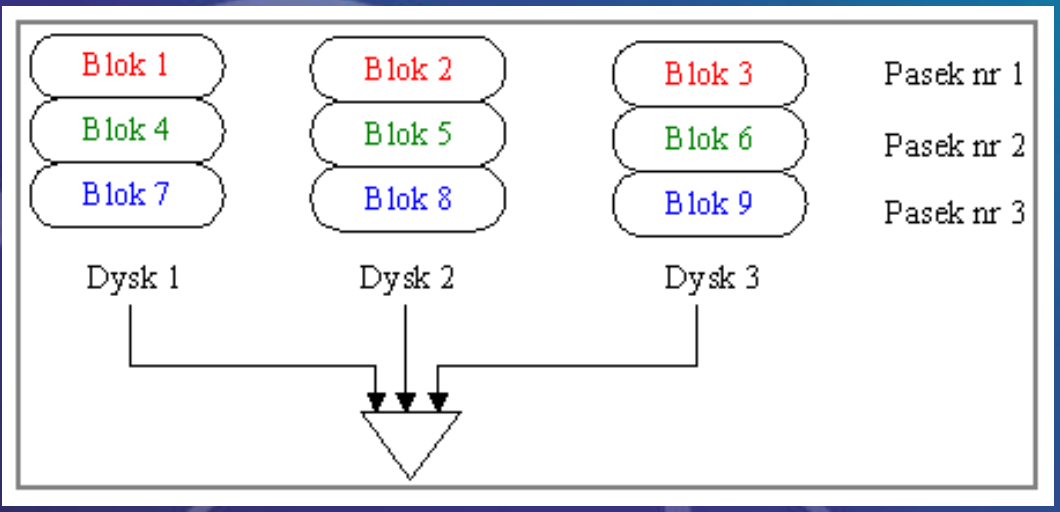
Architektura RAID 0

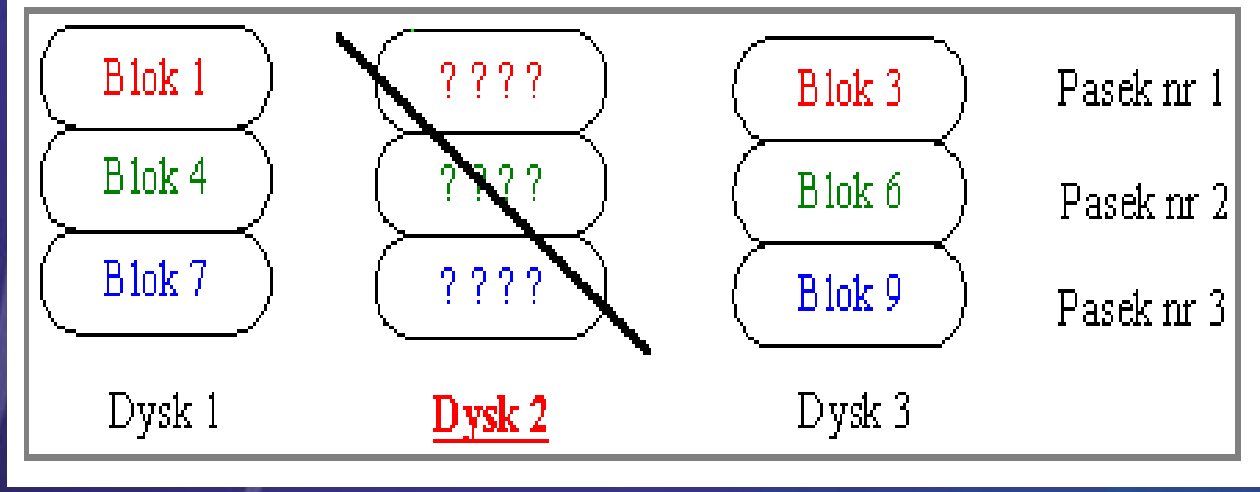
- Dane są podzielone na bloki (najczęściej o rozmiarze 512 bajtów) pomiędzy wszystkie dyski, bez nadmiarowości. Rozwiązanie to jest proste, tanie w implementacji, lecz nie oferuje bezpieczeństwa danych.

- Odczyt danych odbywa się równolegle ze wszystkich dysków.
- RAID 0 znajduje zastosowanie tam, gdzie potrzebna jest bardzo duża wydajność, zarówno w pracy sekwencyjnej, jak i losowej.



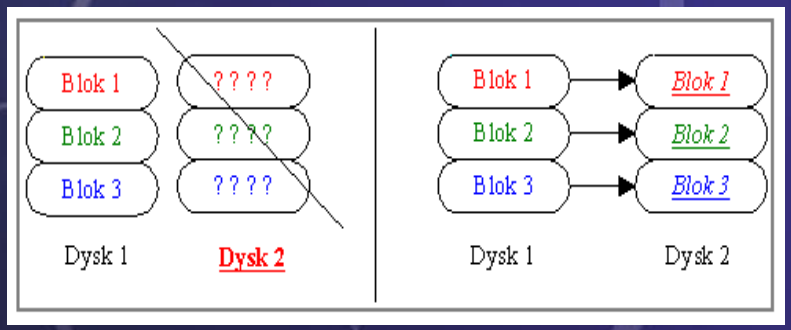
Architektura RAID 0

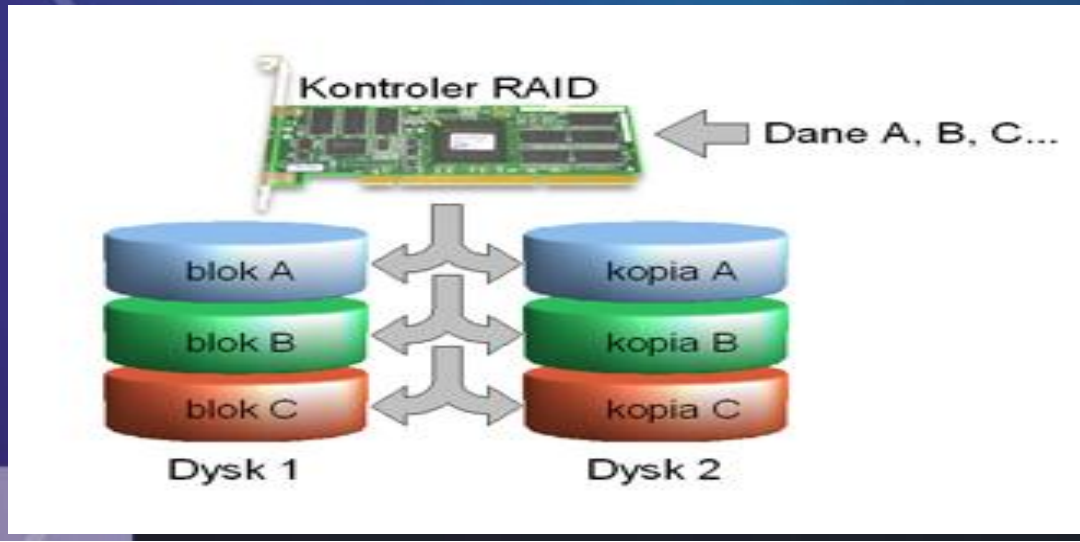




Architektura RAID 1 (Mirroring)

- Realizowany jest zapis danych na dyskach połączonych w pary, dane na obu dyskach w każdej parze są identyczne.
- Wysoki poziom bezpieczeństwa - zaleta.
- Utrata połowy pojemności dysków - wada.
- Jedyny poziomem RAID, który może zapewnić bezpieczeństwo.
- Wykorzystując jedynie 2 dyski - pozostałe wymagają przynajmniej trzech.

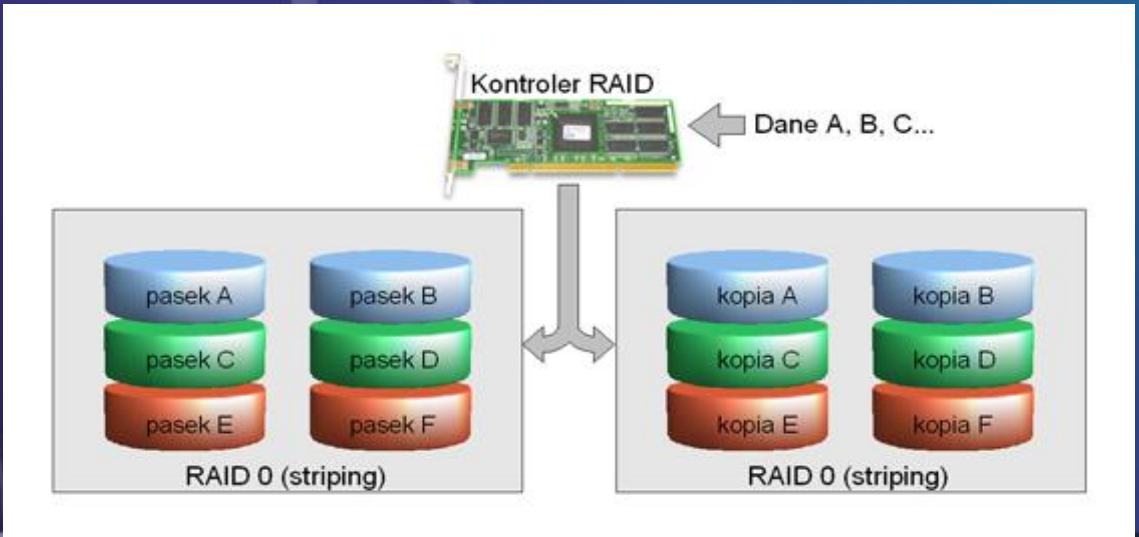




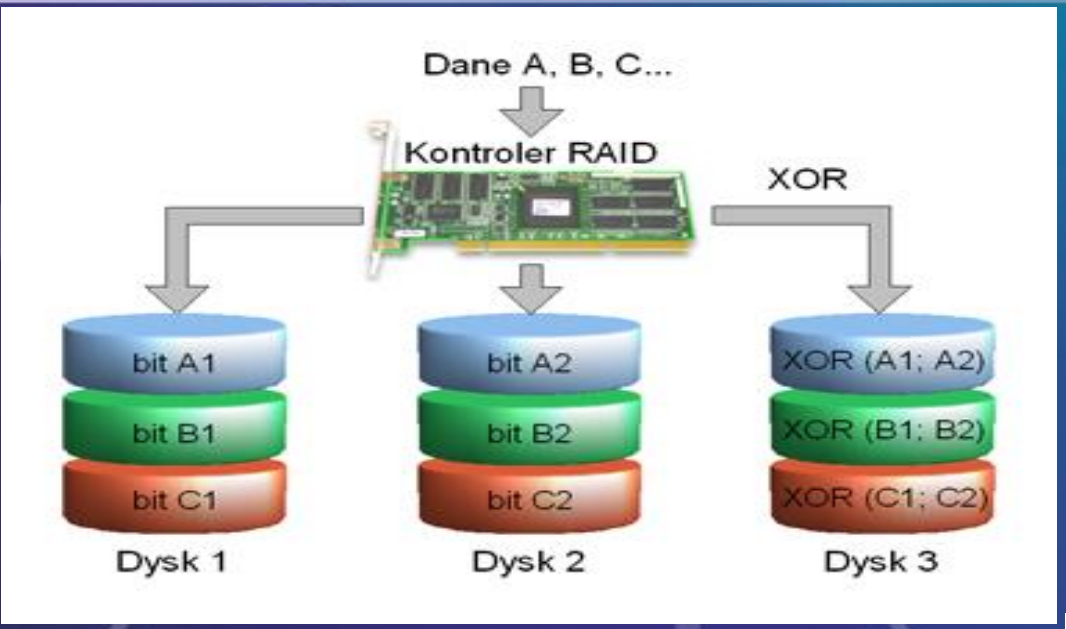
autor: dr inż. Stanisława Plichta

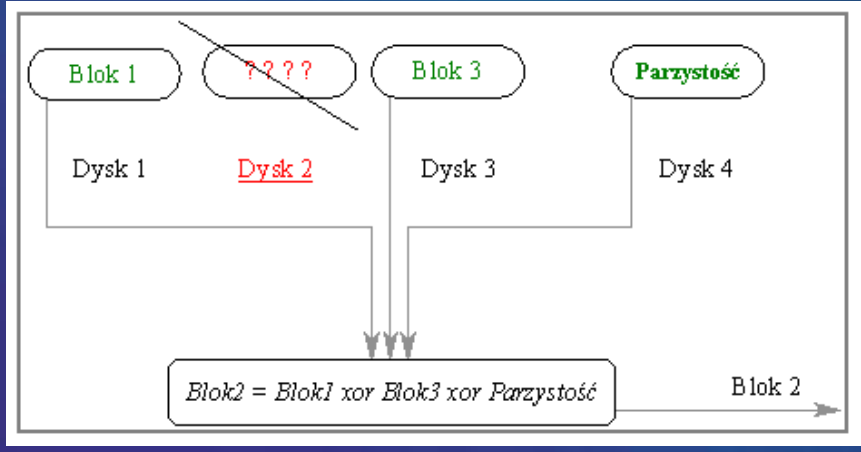
Architektura RAID 0+1

- Połączenie RAID 0 i RAID 1.
- Łączy zalety RAID o i RAID 1 - wydajność i bezpieczeństwo.
- Wykorzystanie połowy pojemności dyskowej – wada z RAID 1.
- Duża przestrzeń dyskową w jednej logicznej całości - najwyższą wydajność przy najwyższych kosztach macierzy.



Architektura RAID 3





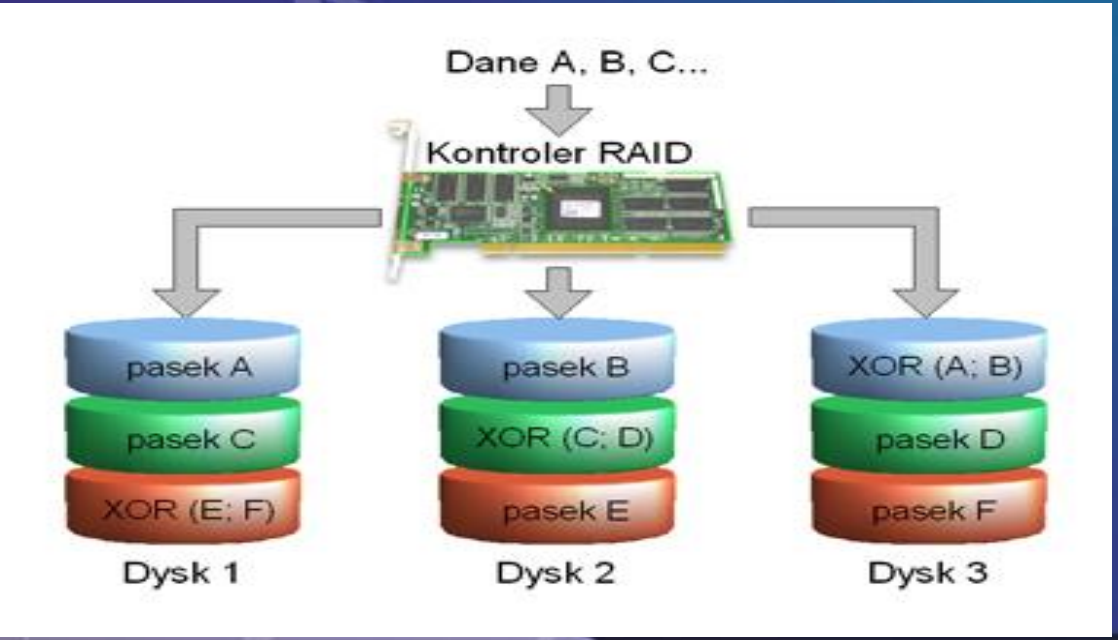
autor: dr inż. Stanisława Plichta

Architektura RAID 3

- Dane zapisywane na kilku dyskach - jeden dysk jest dyskiem parzystości.
- Wydajność macierzy jest gorsza niż w RAID 0 - wykorzystywana jest większa część przestrzeni dyskowej niż w RAID 1.
- Dyski są nierównomiernie obciążone - wada - dysk zapisujący informacje o parzystości jest obciążony znacznie bardziej niż pozostałe.

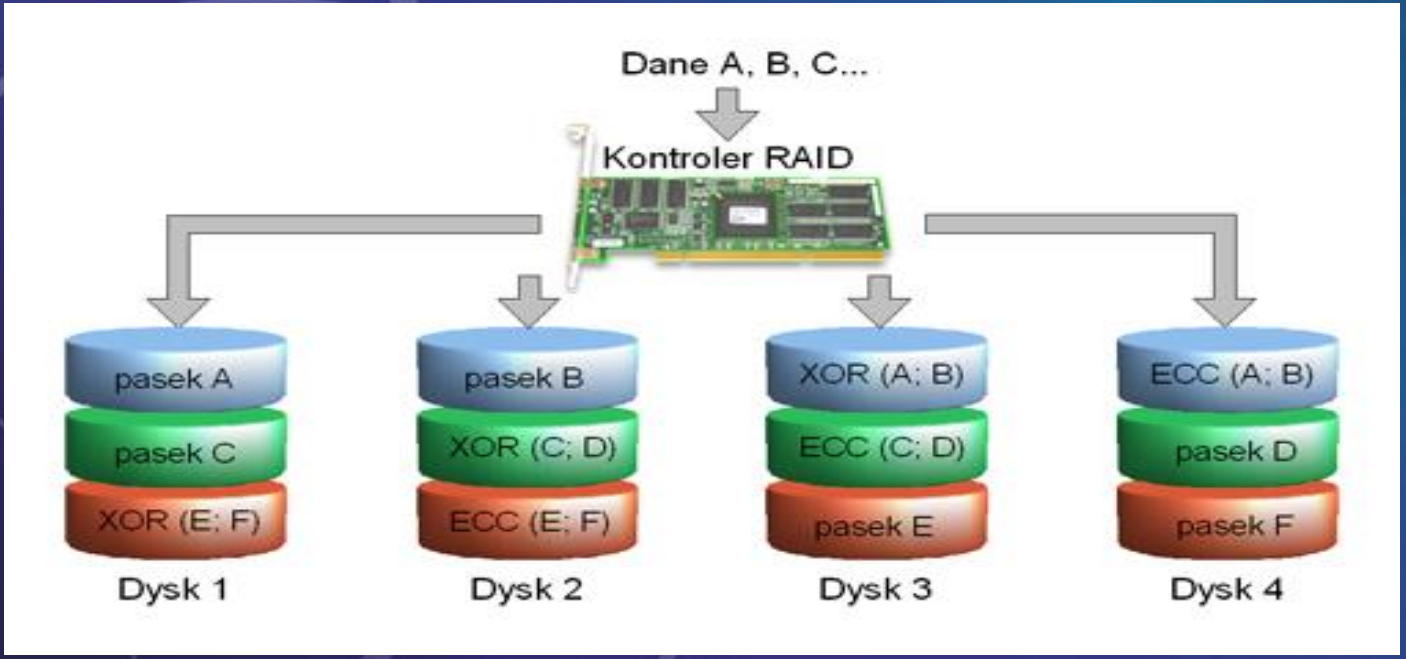
Architektura RAID 5

- Zapis danych jest realizowany na kilku dyskach jednocześnie - rotacyjny zapisem kodów parzystości na wszystkich dyskach pracujących w macierzy.
- Dyski są równomiernie obciążone.
- Zapewnia bezpieczeństwo danych i efektywne wykorzystanie powierzchni nośnika - najczęściej stosowany poziom RAID.



Architektura RAID 6

- Przypomina RAID 5 - przechowuje dodatkową informację nadmiarową na wypadek zwielokrotnionych awarii dysków.
- Zamiast parzystości stosuje kody korygujące.
- Na każde 4 bity danych pamięta 2 bity danych nadmiarowych.



Procesy - UNIX

- Najważniejszą częścią jądra systemu jest podsystem zarządzania procesami.
- Proces to ciąg czynności wykonywanych za pośrednictwem ciągu rozkazów, których wynikiem jest wykonanie pewnych zadań.
- Pod pojęciem procesu możemy rozumieć program (pasywny), który się aktualnie wykonuje (aktywny).
- Proces potrzebuje pewnych zasobów, do których należą:
  - pamięć,
  - procesor,
  - wszelkiego typu urządzenia zewnętrzne.

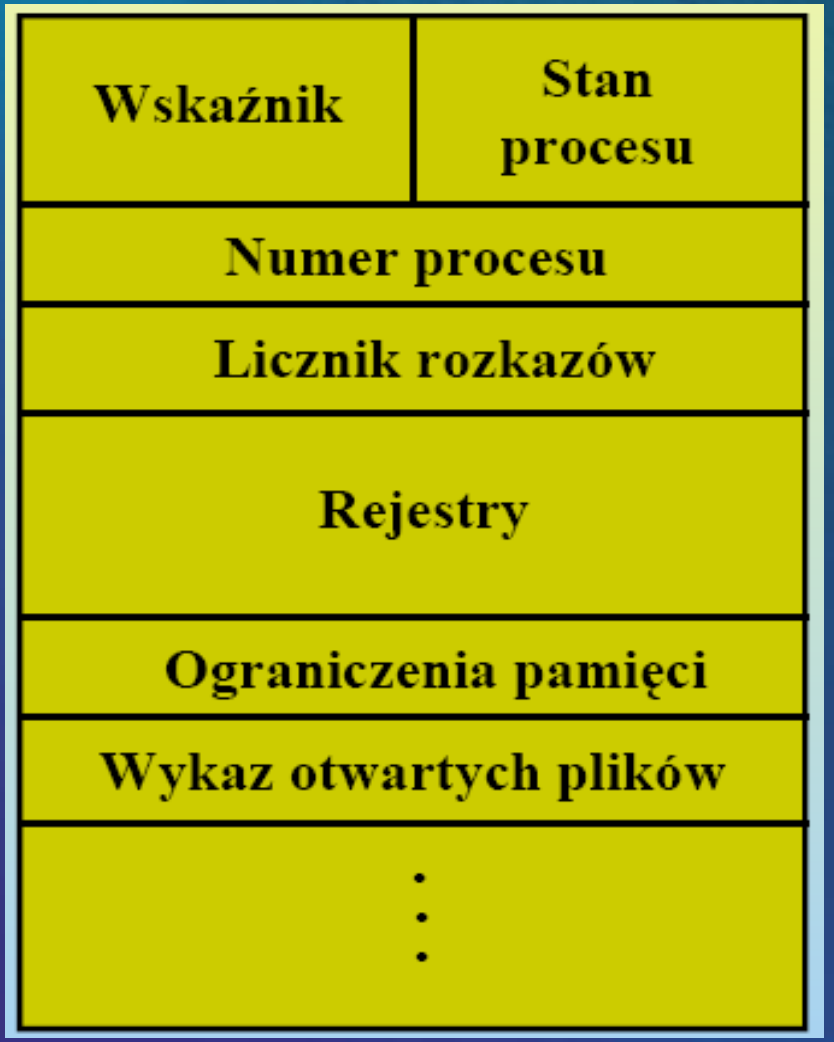
Stany procesu

- Każdy proces otrzymuje jednoznacznie go identyfikujący numer tzw. PID (proces ID).
- Wykonujący się proces zmienia swój stan.
- Stan procesu jest po części określony przez bieżącą czynność procesu.

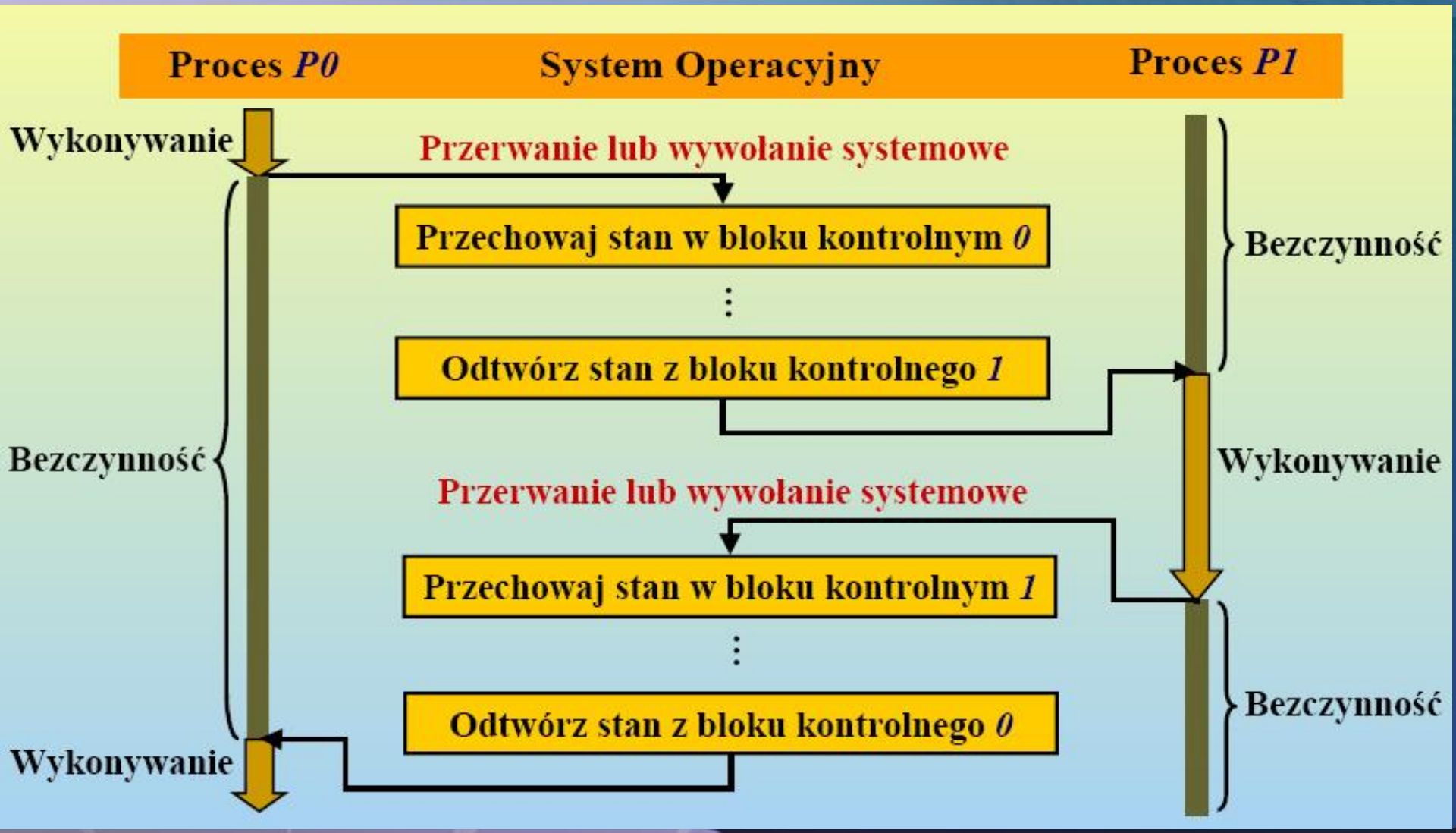
- **AKTYWNY** - są wykonywane instrukcje
- **CZEKAJĄCY** - proces czeka na wystąpienie jakiegoś zdarzenia
- **GOTOWY** - proces czeka na przydział procesora

Blok kontrolny procesu

- SO aby zaspokoić wszystkie potrzeby procesów musi posiadać zbiór informacji o każdym procesie.
- Każdy proces jest reprezentowany w systemie operacyjnym przez swój *blok kontrolny procesu*.



Przełączanie procesów

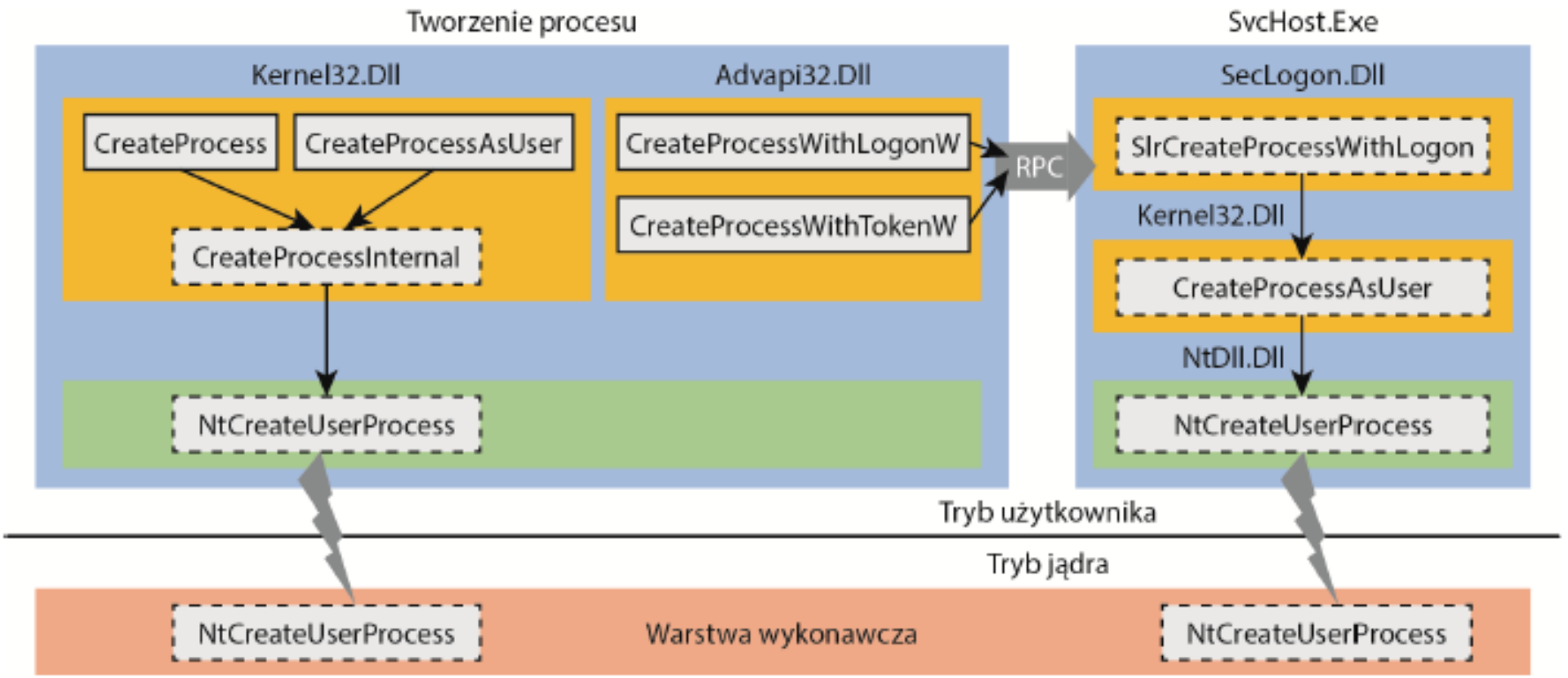


autor: dr inż. Stanisława Plichta

Planowanie procesów - UNIX

- **Planista długoterminowy – planista zadań** *(long term scheduler*
- **Planista krótkoterminowy** *(short term scheduler*
- **Planista średnioterminowy** (*medium term scheduler)*

Procesy – funkcje tworzące procesy



Argumenty funkcji CreateProcess

- Dla funkcji CreateProcessAsUser i CreateProcessWithTokenW argumentem jest dojście tokenu, w ramach którego powinien zostać wykonany nowy proces.
- W przypadku funkcji CreateProcessWithLogonW wymagana jest:
  - nazwa użytkownika, domena i hasło,
  - ścieżka pliku wykonywalnego i argumenty wiersza poleceń.
  - opcjonalne atrybuty zabezpieczeń stosowane dla nowego procesu oraz obiekt wątku, który zostanie utworzony.
  - Flaga wskazująca, czy wszystkie dojścia w bieżącym (tworzącym) procesie oznaczone jako dziedziczne powinny być dziedziczone (kopiowane) do nowego procesu oraz różne flagi wpływające na tworzenie procesu.

Tworzenie procesów

- Każdy proces systemu Windows jest reprezentowany przez strukturę wykonawczą procesu EPROCESS.
- Oprócz wielu atrybutów związanych z procesem, struktura ta zawiera kilka innych powiązanych struktur danych.
- Każdy proces ma jeden lub więcej wątków, z których każdy reprezentowany jest przez strukturę wykonawczą wątku ETHREAD.

Tworzenie innych rodzajów procesów

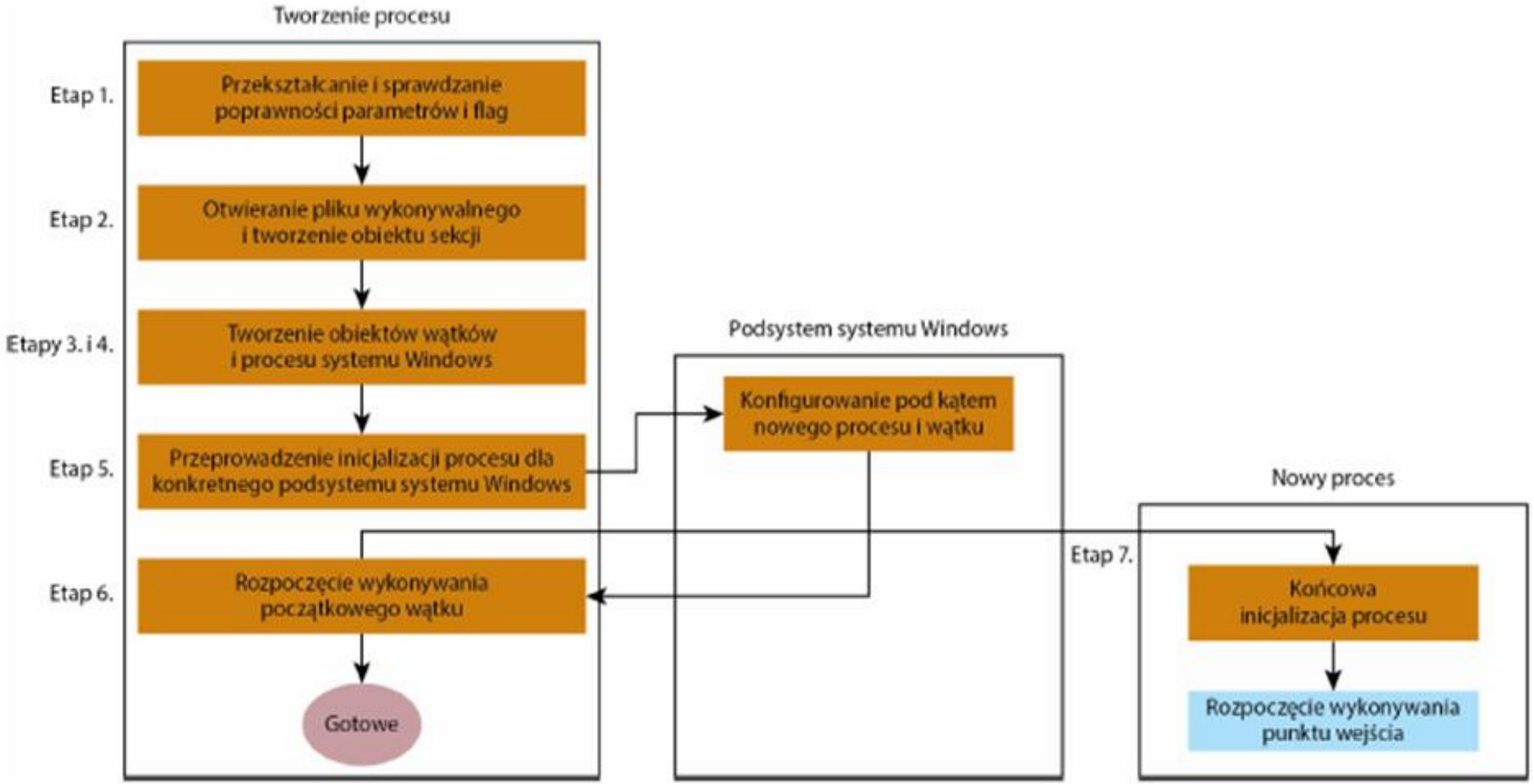
- Warstwa wykonawcza zapewnia obsługę dodatkowych rodzajów procesów, które uruchamiane są z pominięciem interfejsu API systemu Windows:
  - procesy natywne,
  - procesy minimalne,
  - procesy Pico.

Procesy systemowe

Procesy systemowe występujące w każdej odmianie systemu Windows 10

- Proces bezczynności.
- Proces *System.*
- Proces bezpieczeństwa systemu.
- Proces kompresji pamięci.
- Menedżer sesji (*Smss.exe*).
- Podsystem Windows (*Csrss.exe*).
- Sesja 0 inicjalizacji (*Wininit.exe*).
- Proces logowania (*Winlogon.exe*).
- Menedżer kontroli usług (*Services.exe*)
- Usługa uwierzytelniania zabezpieczeń lokalnych (*Lsass.exe*).

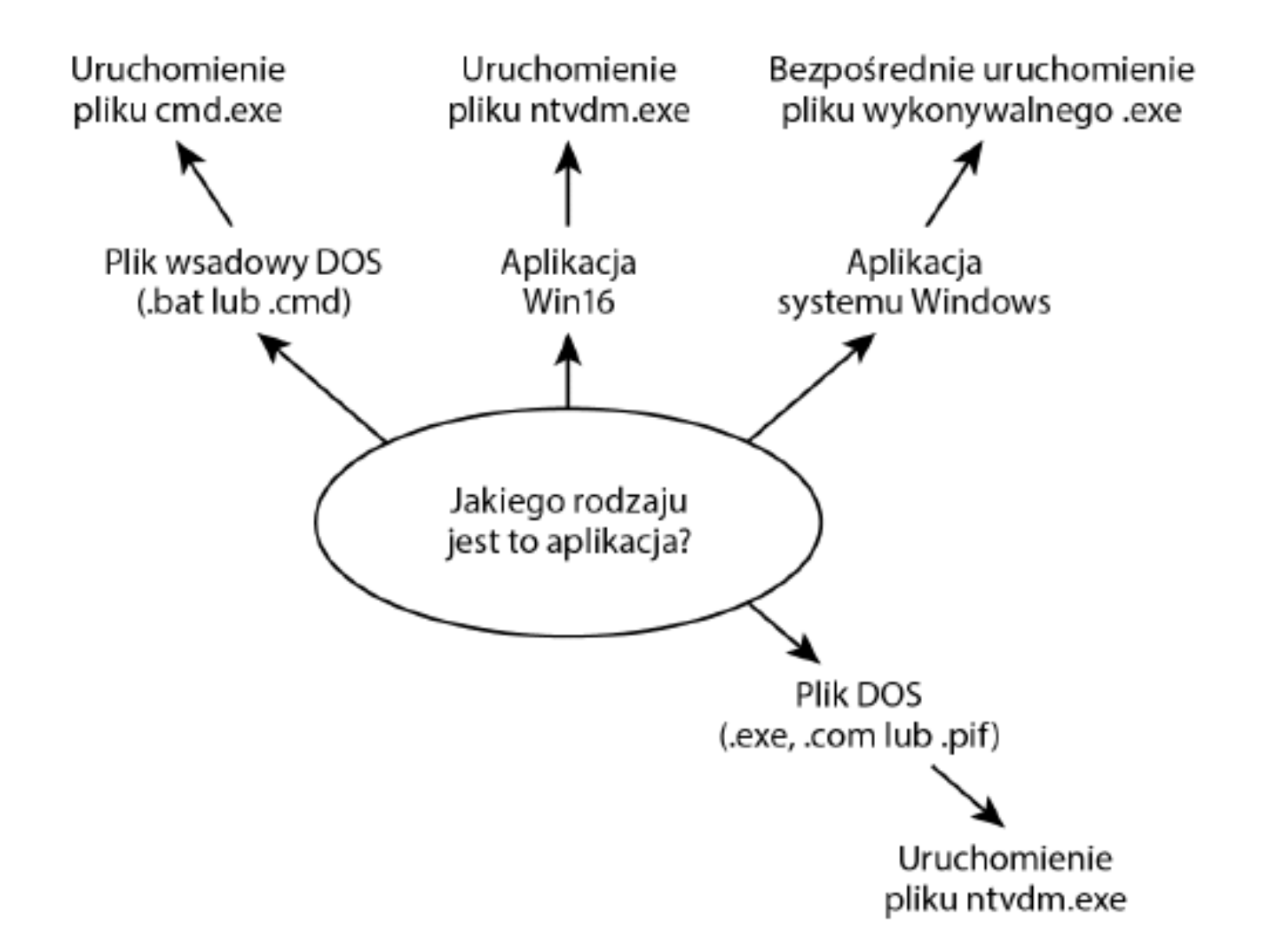
Główne etapy tworzenia procesu



Etap 1 - Przekształcanie i sprawdzanie poprawności parametrów i flag

- Klasa priorytetu nowego procesu określana jest jako niezależne bity w parametrze CreationFlags funkcji CreateProcess*.
- Zdefiniowanych jest sześć klas priorytetu procesu - każda wartość mapowana jest na liczbę:
  - *Bezczynny* lub *Niski* (4).
  - *Poniżej normalnego* (6).
  - *Normalny* (8).
  - *Powyżej normalnego* (10).
  - *Wysoki* (13).
  - *Czasu rzeczywistego* (24).
- Klasa priorytetu pełni rolę priorytetu bazowego w wypadku wątków tworzonych w danym procesie.
- Jeśli dla nowego procesu nie określono żadnej klasy priorytetu, domyślnie zostanie użyta klasa *Normalny*.

Etap 2 - Otwieranie obrazu do wykonania



Etap 3 - Tworzenie obiektu procesu wykonawczego systemu Windows

- Funkcja NtCreateUserProcess otwiera właściwy plik wykonywalny systemu Windows i tworzy obiekt sekcji.
- Tworzy obiekt procesu wykonawczego systemu Windows - polega to na utworzeniu wątku i obejmuje następujące podetapy:
  - Skonfigurowanie obiektu EPROCESS.
  - Utworzenie początkowego obszaru adresów procesu.
  - Zainicjowanie struktury procesu jądra (KPROCESS).
  - Zakończenie konfigurowania obszaru adresów procesu.
  - Skonfigurowanie struktury PEB (*Process Environment Block*).
  - Zakończenie konfigurowania obiektu procesu wykonawczego.

Etap 4 - Tworzenie początkowego wątku oraz jego stosu i kontekstu

- Początkowy wątek jest tworzony wewnętrznie przez jądro bez udziału trybu użytkownika.
- Funkcja PspAllocateThread obsługuje tworzenie i inicjalizowanie obiektu wątku wykonawczego.
- Funkcja PspInsertThread dzięki funkcji KeStartThread przekształca obiekt wykonawczy w możliwy do planowania wątek w systemie.
- Wątek tworzony jest w stanie wstrzymania i nie jest wznawiany do momentu całkowitego zainicjowania procesu.

Etap 5 - Przeprowadzanie inicjalizacji powiązanej z podsystemem systemu Windows

- Aby zakończyć inicjalizowanie procesu, funkcja CreateProcessInternalW wykonuje różne działania powiązane z operacjami dotyczącymi podsystemu systemu Windows.
- Przeprowadza różne sprawdzenia w celu stwierdzenia, czy system Windows powinien zezwolić na uruchomienie pliku wykonywalnego.
- Na podstawie zebranych informacji tworzony jest komunikat kierowany do podsystemu systemu Windows.

Etap 6 - Rozpoczęcie wykonywania wątku początkowego

- Na tym etapie określa się środowisko procesu i przydziela zasoby dla jego wątków.
- Proces zawiera wątek, a podsystem systemu Windows dysponuje informacją o nowym procesie.
- Jeśli obiekt wywołujący nie określił flagi CREATE_SUSPENDED, wznawiany jest wątek początkowy.

Kończenie procesu

- Proces może zakończyć działanie w poprawny sposób przez wywołanie funkcji ExitProcess przez sam proces żądający zakończenia swojego działania.
- Zakończenie procesu w niepoprawny sposób jest możliwe przy użyciu funkcji TerminateProcess, która może zostać wywołana poza procesem.
- Nigdy nie mogą występować „wycieki - cała pamięć prywatna procesu jest automatycznie zwalniana przez jądro, usuwany jest obszar adresów, zamykane są wszystkie dojścia do obiektów jądra.

Wątki - UNIX

- Wątek *(thread)*, zwany także procesem lekkim *(light-weight process)*, jest podstawową jednostką wykorzystania CPU.
- Posiada: licznik rozkazów, zbiór rejestrów i obszar stosu.
- Wątek dzieli wraz z innymi równorzędnymi wątkami: sekcję kodu, sekcję danych oraz zasoby systemowe (otwarte pliki, sygnały itd.)
- Tradycyjny proces, tzw. ciężki *(heavy-weight)*, jest równoważny zadaniu z jednym wątkiem.

Wątki (threads)

Zalety wątków:

- Dzielenie zasobów sprawia, ze przełączanie między wątkami oraz tworzenie wątków jest tanie w porównaniu z procesami ciężkimi.
- Oszczędne wykorzystanie zasobów – dzięki ich współużytkowaniu.
- Współpraca wielu wątków pozwala zwiększyć przepustowość i poprawić wydajność (np. jeśli jeden wątek jest zablokowany, to może działać inny).
- Wykorzystanie architektury wieloprocesorowej (wątek → procesor).

Utworzenie wątku

int pthread_create(pthread_t *tid , const pthread_attr_t *attr,

void *(*func)(void *), void *arg)

- Do każdego wątku odnosimy się za pośrednictwem identyfikatora wątku,
- Każdy wątek ma wiele atrybutów, które przy jego tworzeniu można określić – zazwyczaj korzysta się z wartości domyślnych przekazując wskaźnik pusty w miejsce argumentu attr

Wątek może pobrać wartość własnego identyfikatora

pthread_t pthread_self(void)

(odpowiednik funkcji getpid())

Przyłączenie i odłączenie wątku

przyłączenie wątku

int pthread_join(pthread_t tid,void **status)

odłączenie wątku

int pthread_detach(pthread_t tid)

Tworzenie wątków - Windows

CreateThread

tworzy wątek w bieżącym procesie

- Akceptuje następujące argumenty:
  - Opcjonalna struktura atrybutów zabezpieczeń.
  - Opcjonalna wielkość stosu.
  - Wskaźnik funkcji.
  - Opcjonalny argument.
  - Opcjonalne flagi.

Tworzenie wątków

CreateRemoteThread
rozszerzona funkcja tworzenia wątku

- Za pomocą tej funkcji możesz umieścić wątek w innym procesie.
- Typowym zastosowaniem tej techniki jest wymuszanie przerwania w debugowanym procesie.

Tworzenie wątków

CreateRemoteThreadEx

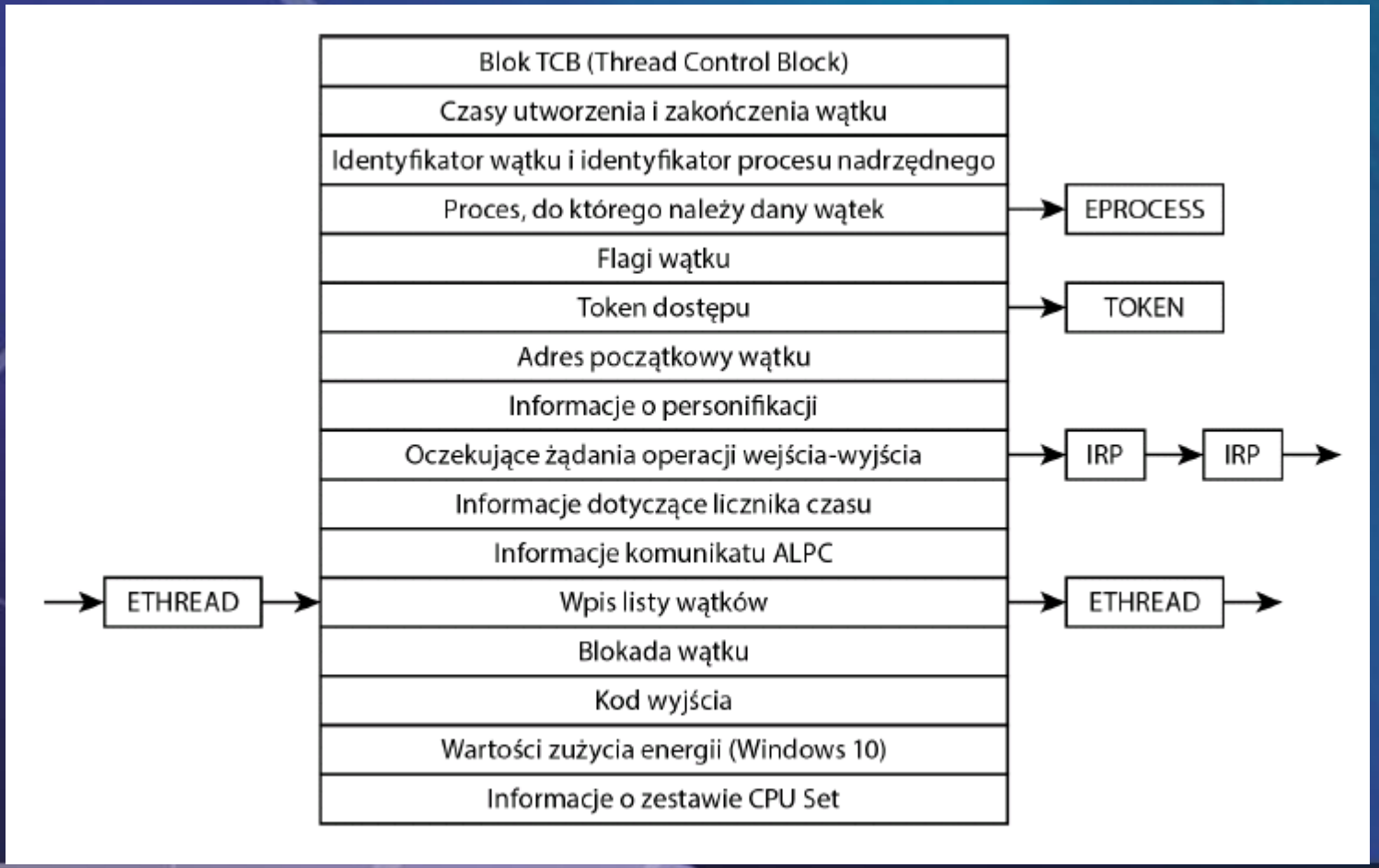
nadzbiór funkcji CreateThread i CreateRemoteThread.

PsCreateSystemThread
tworzenie wątku w trybie jądra

przydatne w wypadku sterowników, które wymagają, aby niezależne działania miały postać procesów w obrębie procesu systemowego.

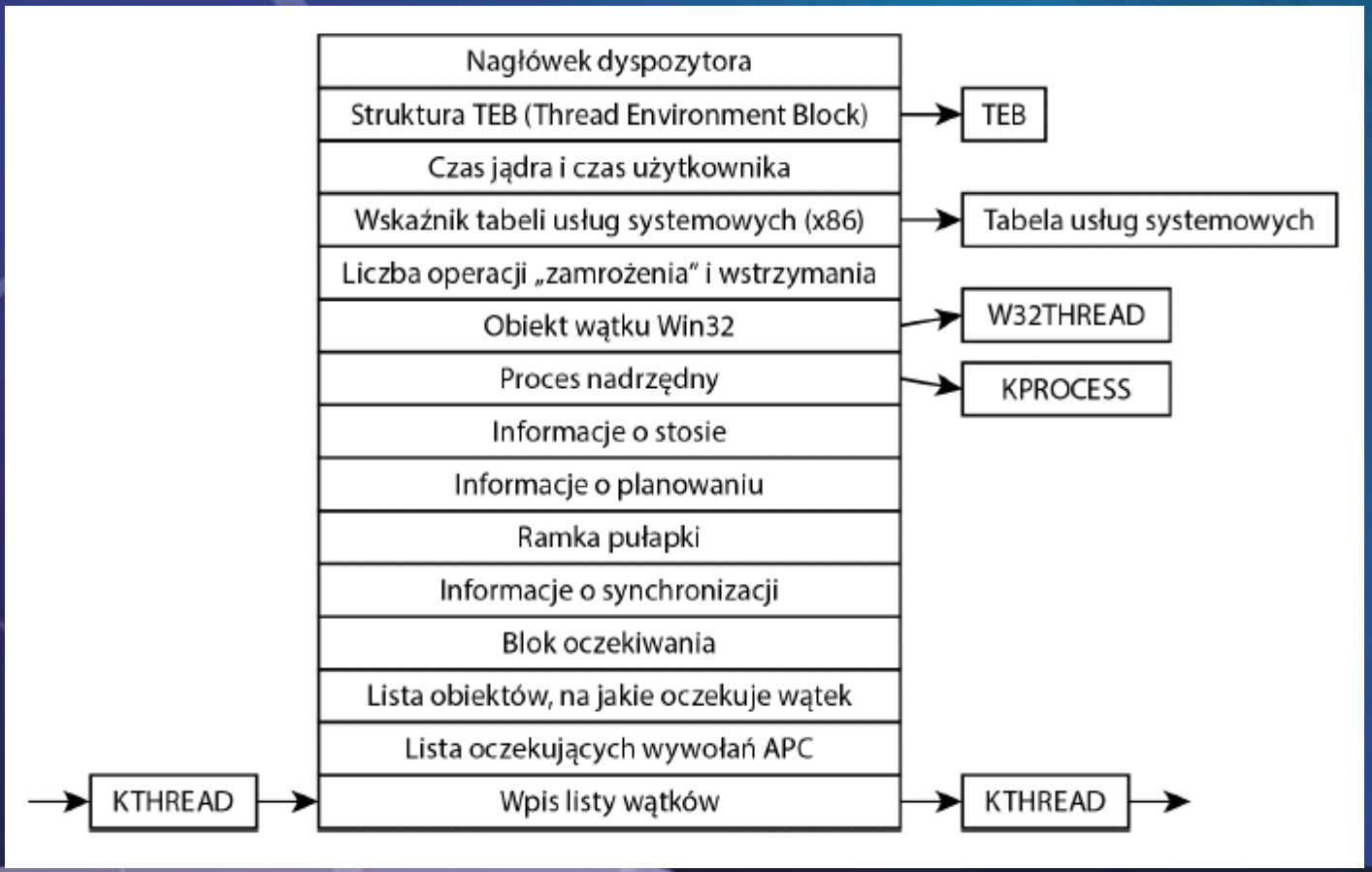
Struktury danych

Pola wykonawczej struktury wątku (ETHREAD)



Struktury danych

Pola struktury wątku jądra (KTHREAD)



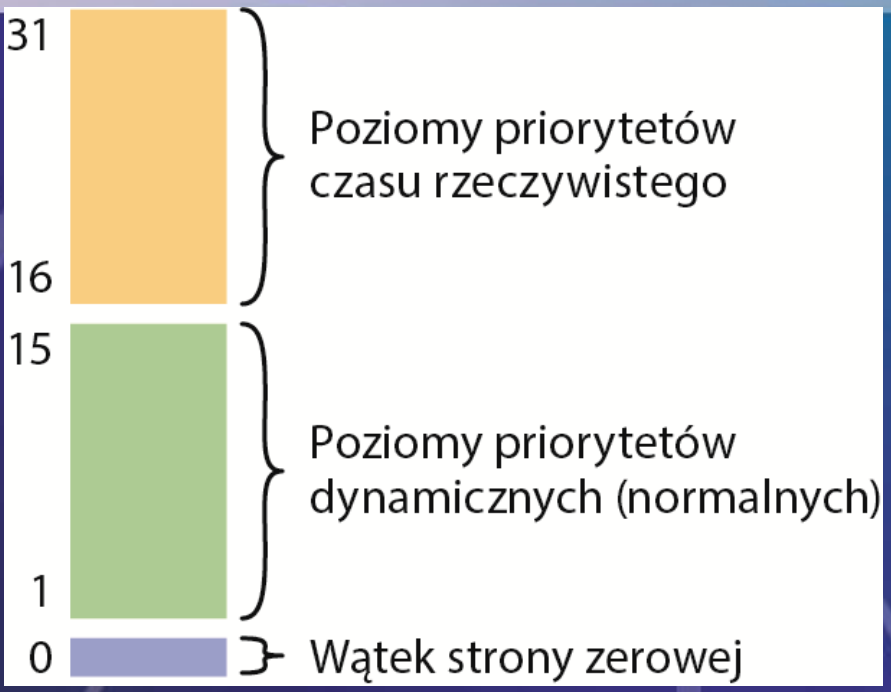
Planowanie w systemie Windows

- System Windows implementuje mechanizm wywłaszczeniowego planowania sterowanego za pomocą priorytetów.
- Po wybraniu wątku do uruchomienia działa on przez kwant czasu.
- Wartości kwantu mogą się zmieniać dla poszczególnych systemów i procesów.
- Kod planowania w systemie Windows implementowany jest w jądrze - nie istnieje żaden moduł planowania ani funkcja planowania.

Planowanie w systemie Windows

- Dyspozytor jądra zawiera funkcje odpowiedzialne za obsługę planowania.
- System Windows musi określić, jaki wątek powinien działać jako następny w procesorze logicznym gdy mają miejsce zdarzenia:
  - Wątek staje się gotowy do wykonania — na przykład wątek został właśnie utworzony lub przełączony ze stanu oczekiwania.
  - Upływa kwant czasu, wątek został zakończony, przestaje być wykonywany lub przechodzi w stan oczekiwania.
  - Zmienia się priorytet wątku z powodu wywołania usługi systemowej lub zmiany wartości priorytetu przez sam system Windows.
  - Wątek przestaje działać w procesorze, w którym został uruchomiony.

Poziomy priorytetów



- Czasu rzeczywistego (4)
- Wysoki (3)
- Powyżej normalnego (6)
- Normalny (2)
- Poniżej normalnego (5)
- Bezczynny (1)

wewnętrzny indeks
PROCESS_PRIORITY_CLASS

Poziomy priorytetów wątków są przypisywane z dwóch różnych perspektyw związanych z:

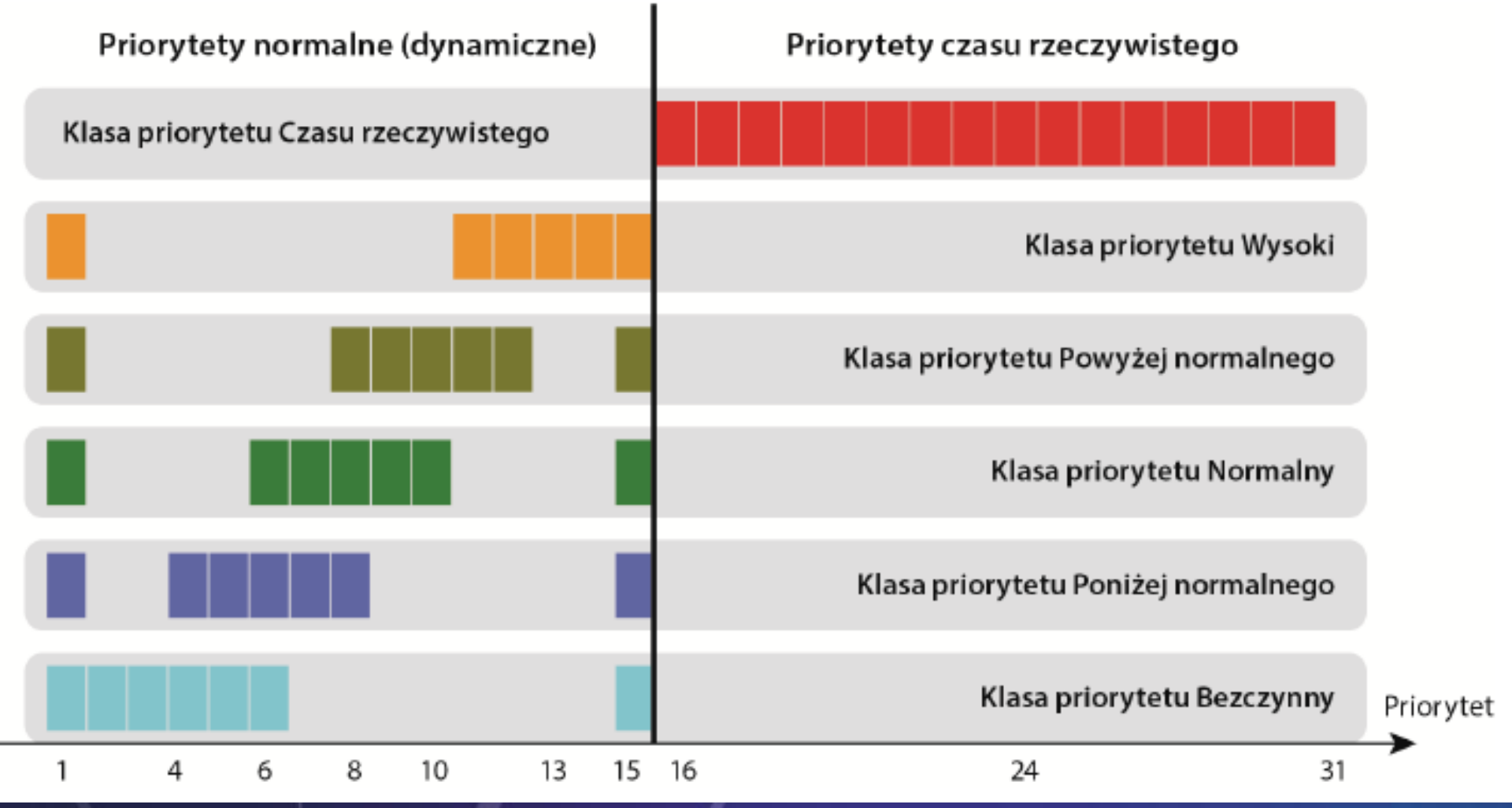
- interfejsem API systemu Windows
- jądrem systemu.

Poziomy priorytetów

- Interfejs API SetPriorityClass systemu Windows umożliwia zmianę klasy priorytetu procesu.
- Interfejs API systemu Windows przypisuje względny priorytet poszczególnych wątków w procesach.
  - Krytyczny pod względem czasu (15).
  - Najwyższy (2).
  - Powyżej normalnego (1).
  - Normalny (0).
  - Poniżej normalnego (–1).
  - Najniższy (–2).
  - Bezczynny (–15).

różnica priorytetu stosowana dla priorytetu bazowego procesu

Mapowanie priorytetów

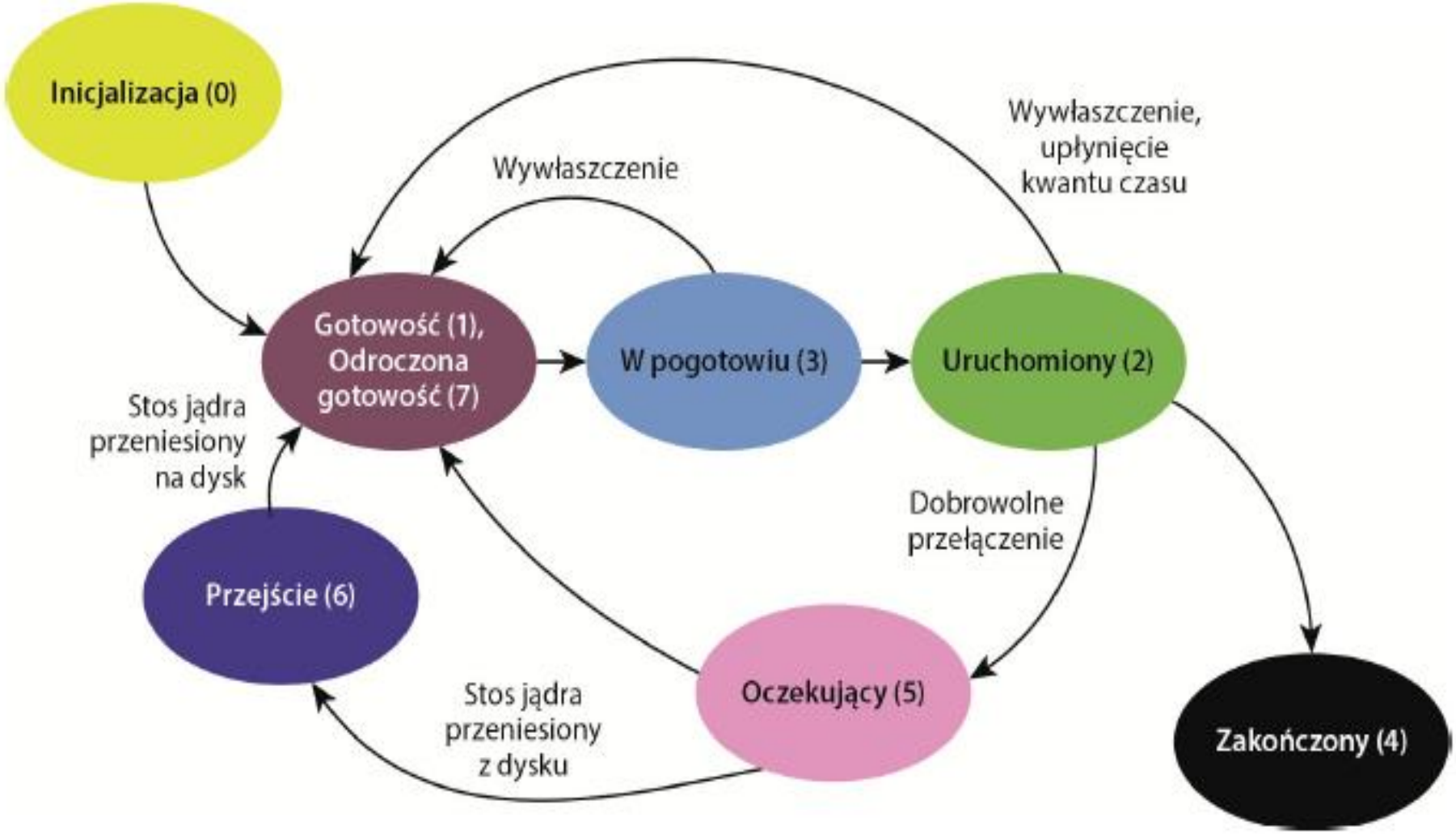


Narzędzia pozwalające na interakcję z priorytetem

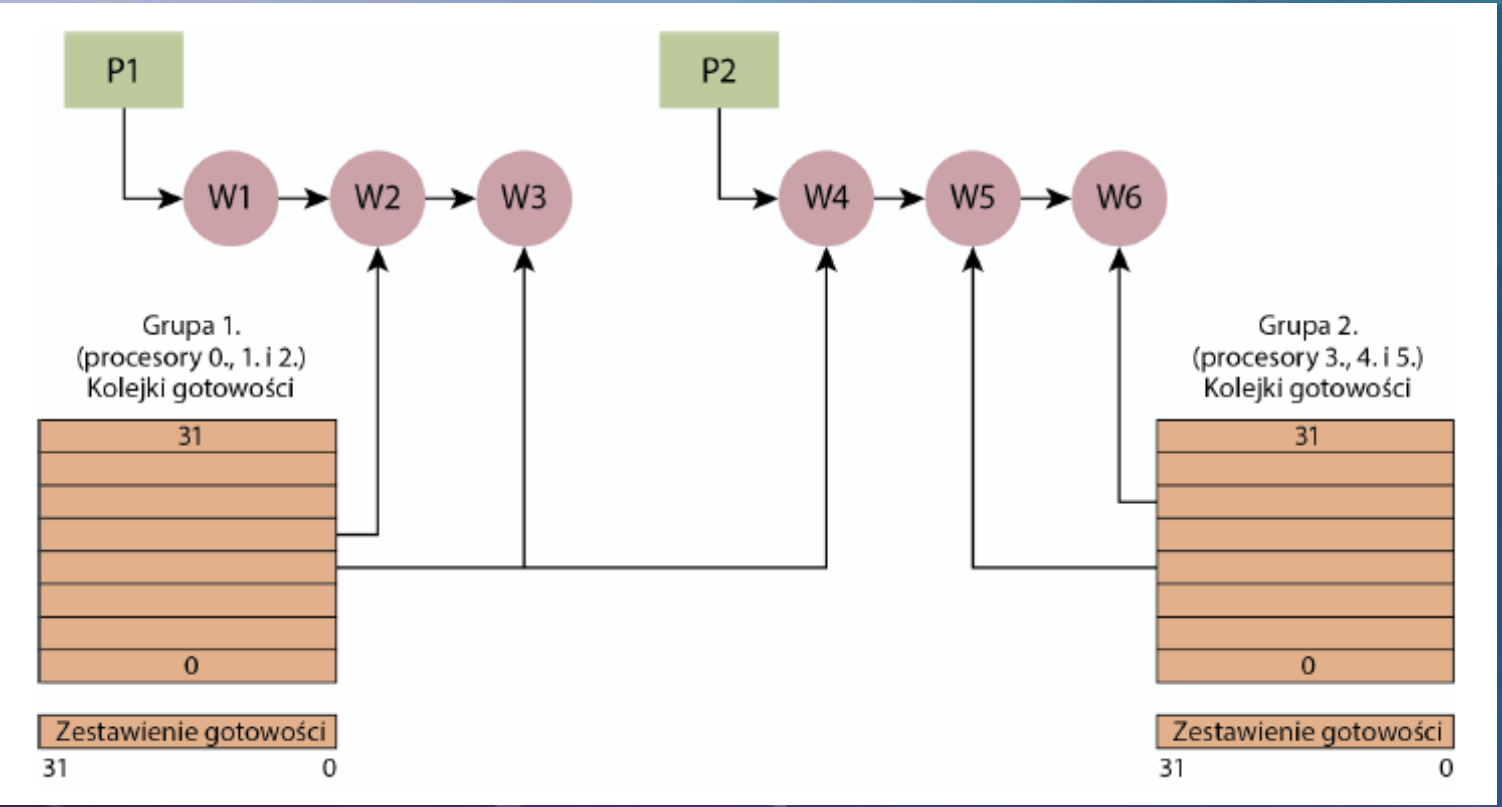
- Menedżer zadań i narzędzie Process Explorer pozwalają zmienić priorytet bazowy procesu.
- Narzędzie Monitor wydajności, Process Explorer lub WinDbg pozwala na wyświetlenie priorytetów poszczególnych wątków.
- Jedynym sposobem określenia początkowej klasy priorytetu dla procesu jest zastosowanie polecenia start z poziomu wiersza poleceń systemu Windows.

cmd/c start /low Notepad.exe.

Stany wątków



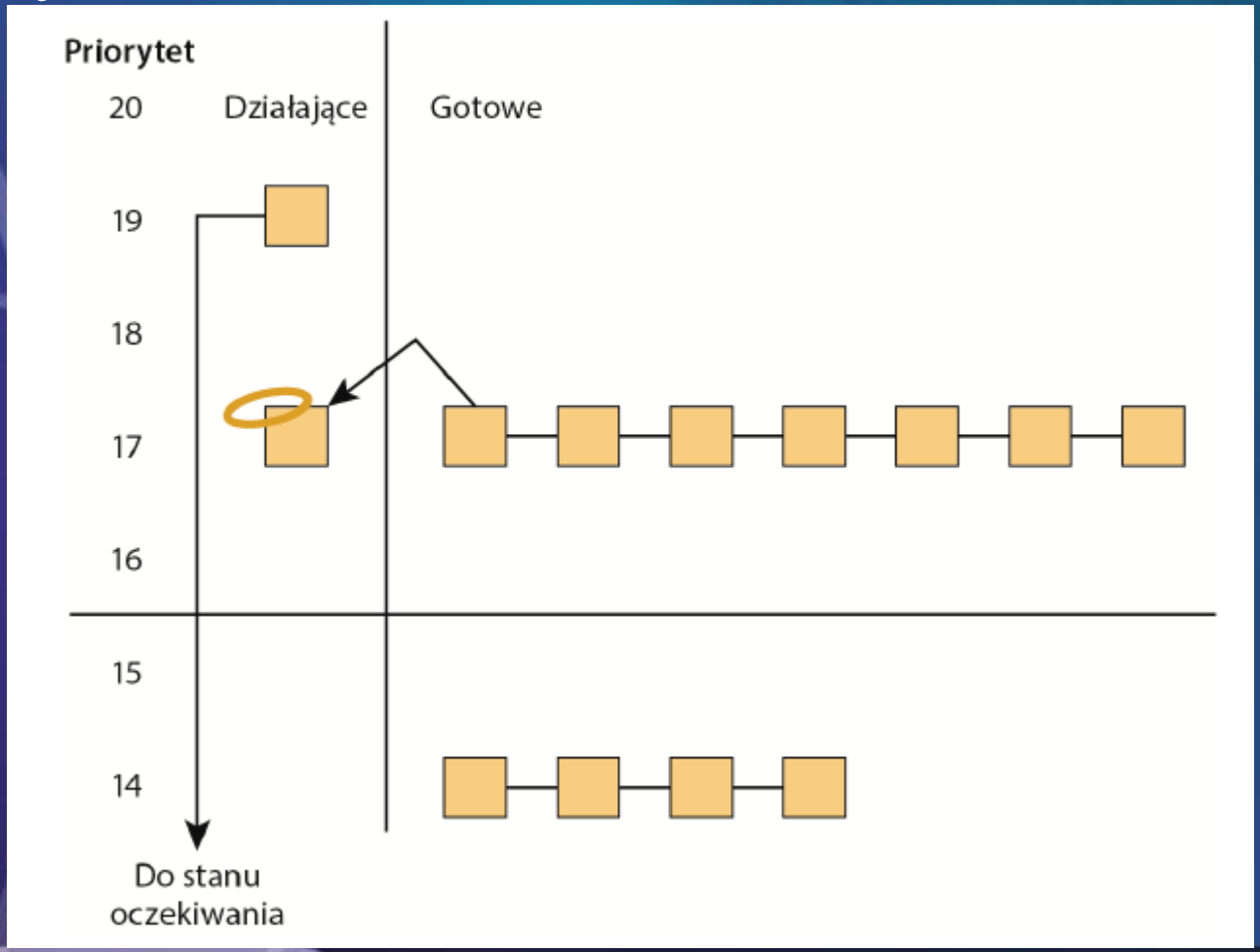
Baza danych dyspozytora



przykład dotyczy sześciu procesorów
P reprezentuje procesy
W identyfikuje wątki

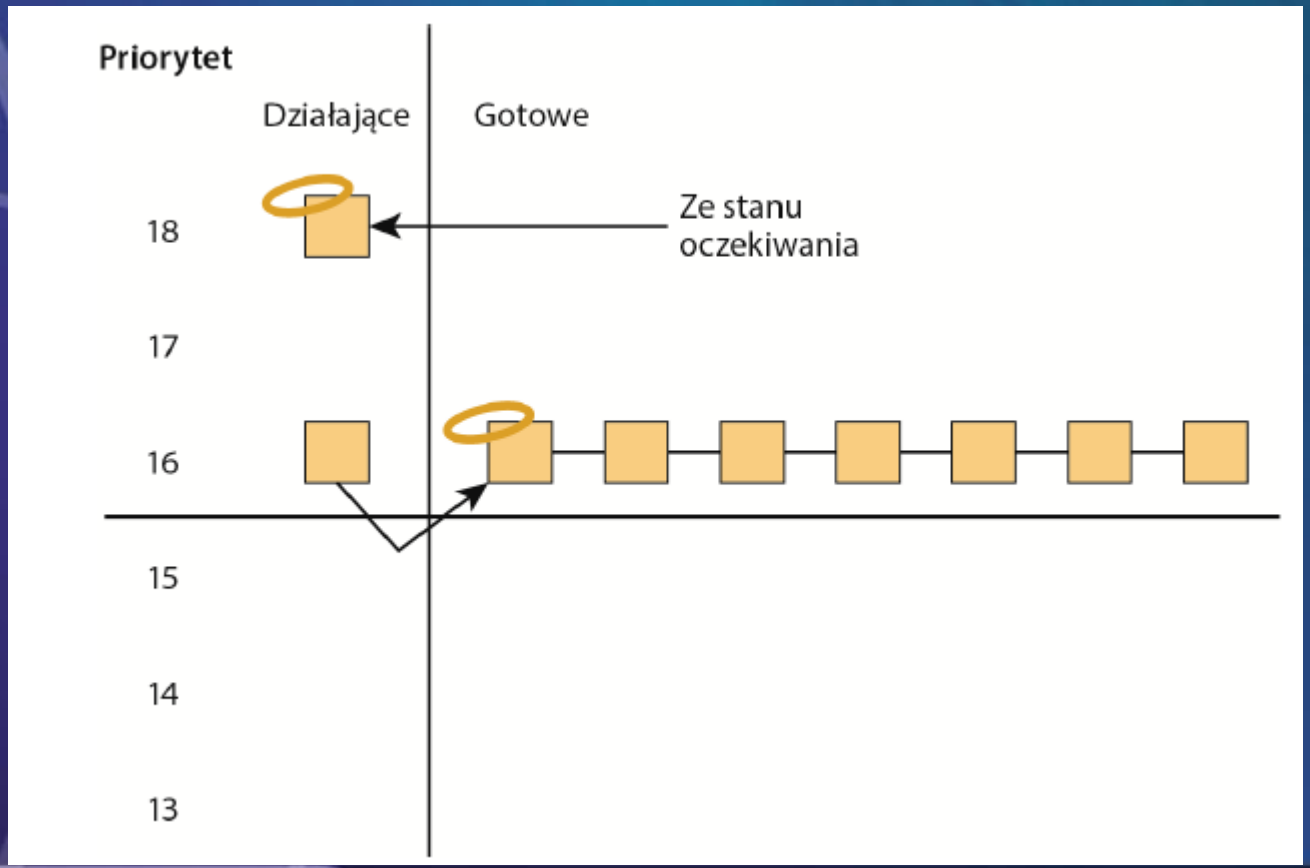
Warianty planowania

Dobrowolne przełączanie



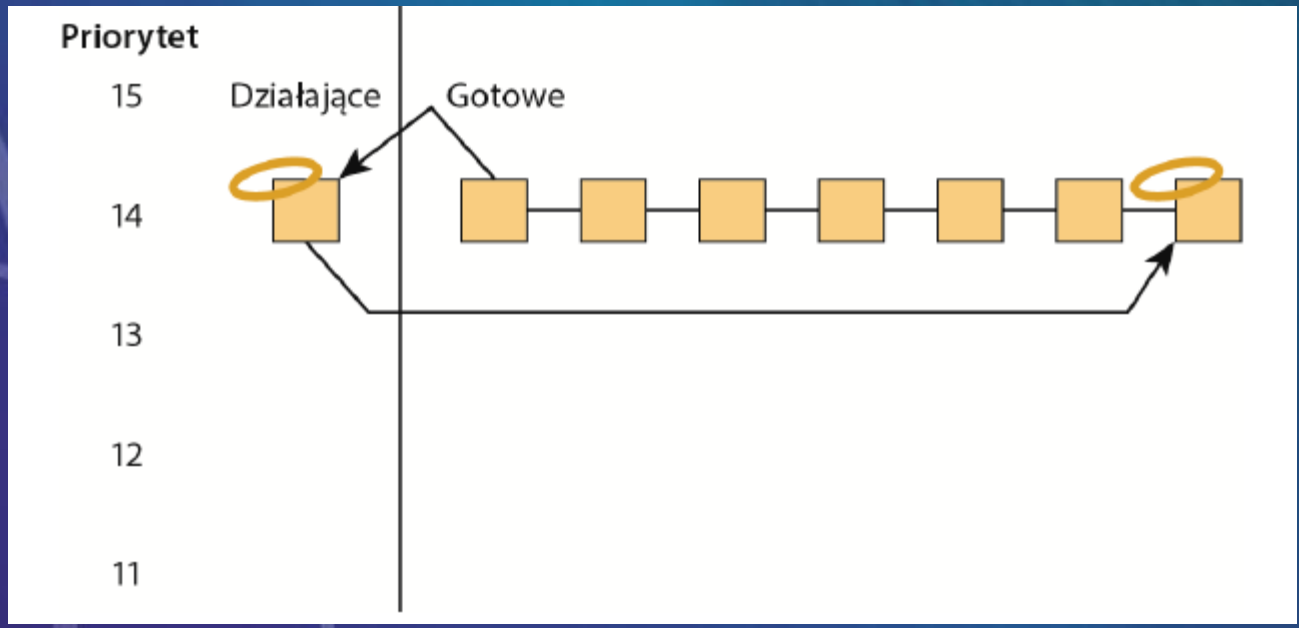
Warianty planowania

Planowanie wątków z wywłaszczaniem



Warianty planowania

Planowanie wątków z upłynięciem kwantu czasu



Wątki działające w trybie użytkownika mogą wywłaszczać wątki uruchomione w trybie jądra. Tryb, w którym działa wątek, nie ma znaczenia - decydującym czynnikiem jest jego priorytet.

Wątek bezczynności

- Gdy w procesorze nie ma żadnego wątku możliwego do uruchomienia, system Windows przekazuje wątek bezczynności procesora.
- Każdy procesor ma własny, dedykowany wątek bezczynności.
- Wszystkie wątki bezczynności należą do procesu bezczynności.

Wybór wątku

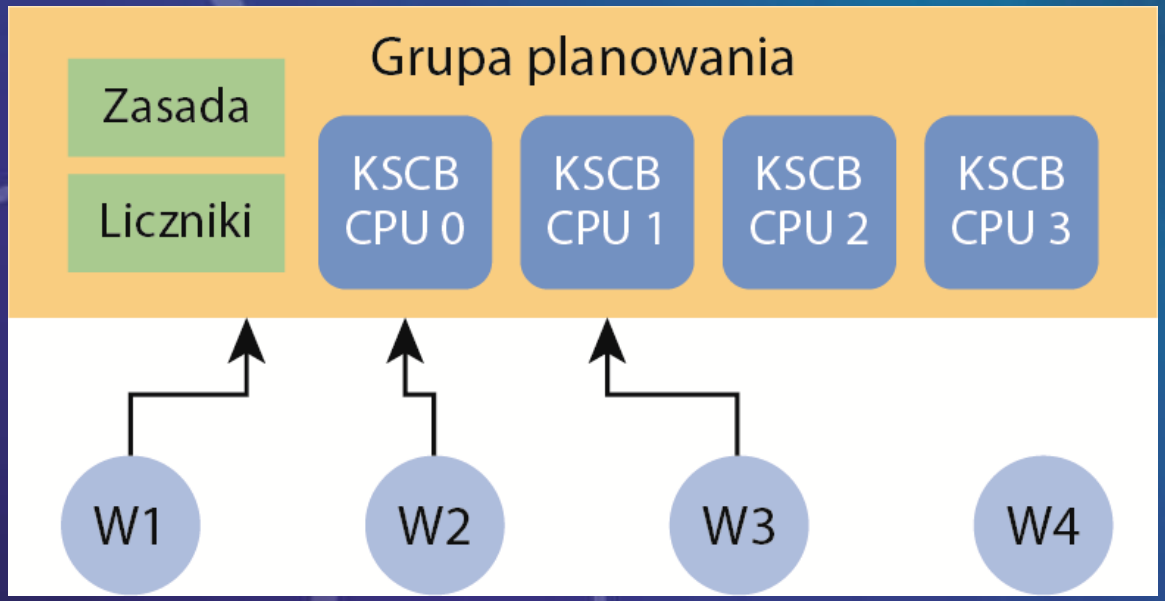
- Gdy procesor logiczny musi wybrać następny wątek do uruchomienia, wywołuje funkcję planisty KiSelectNextThread.

Co robi planista ?

- Planista wywołuje funkcję KiSelectReadyThreadEx w celu wyszukania następnego gotowego wątku, który powinien zostać uruchomiony, jest on przełączany w stan gotowości.
- Jeśli nie znaleziono gotowego wątku, aktywowany jest planista bezczynności i do wykonania wybierany jest wątek bezczynności.

Planowanie oparte na grupach

- Od systemach Windows 8 i Windows Server 2012 wprowadzono mechanizm planowania oparty na grupach.



Wątki W1, W2 i W3 należą do grupy planowania, natomiast wątek W4 nie należy

Ranking - ważny parametr utrzymywany przez grupę planowania

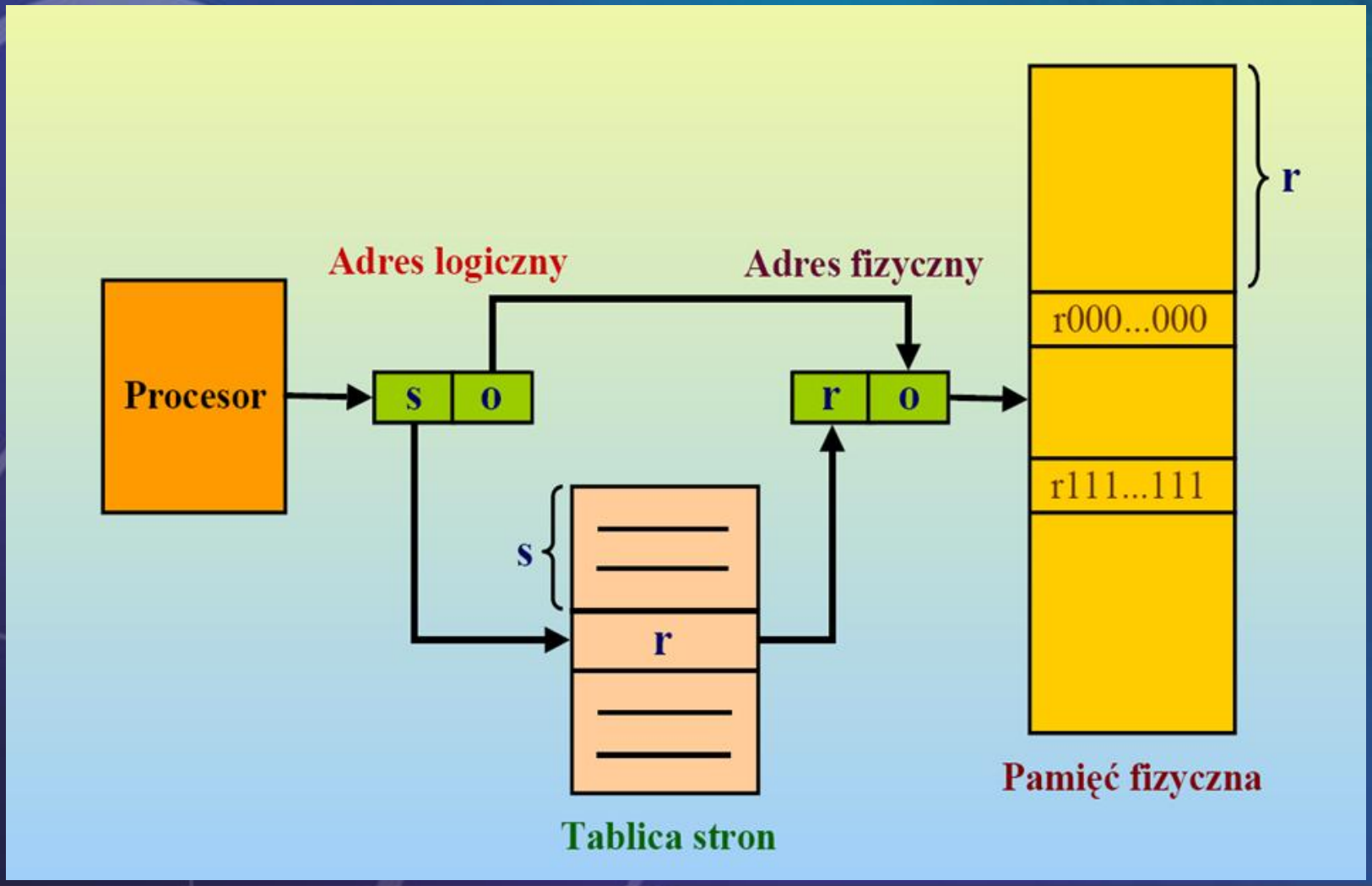
Zarządzanie pamięcią operacyjną

- adres logiczny - wytworzony przez procesor (adres wirtualny)
- zbiór wszystkich adresów logicznych - **logiczna przestrzeń adresowa**
- adres fizyczny - umieszczony w rejestrze adresowym pamięci
- zbiór wszystkich adresów fizycznych - **fizyczna przestrzeń adresowa**
- odwzorowanie adresów wirtualnych na fizyczne - MMU (jednostka zarządzająca *pamięcią –memory management unit*)

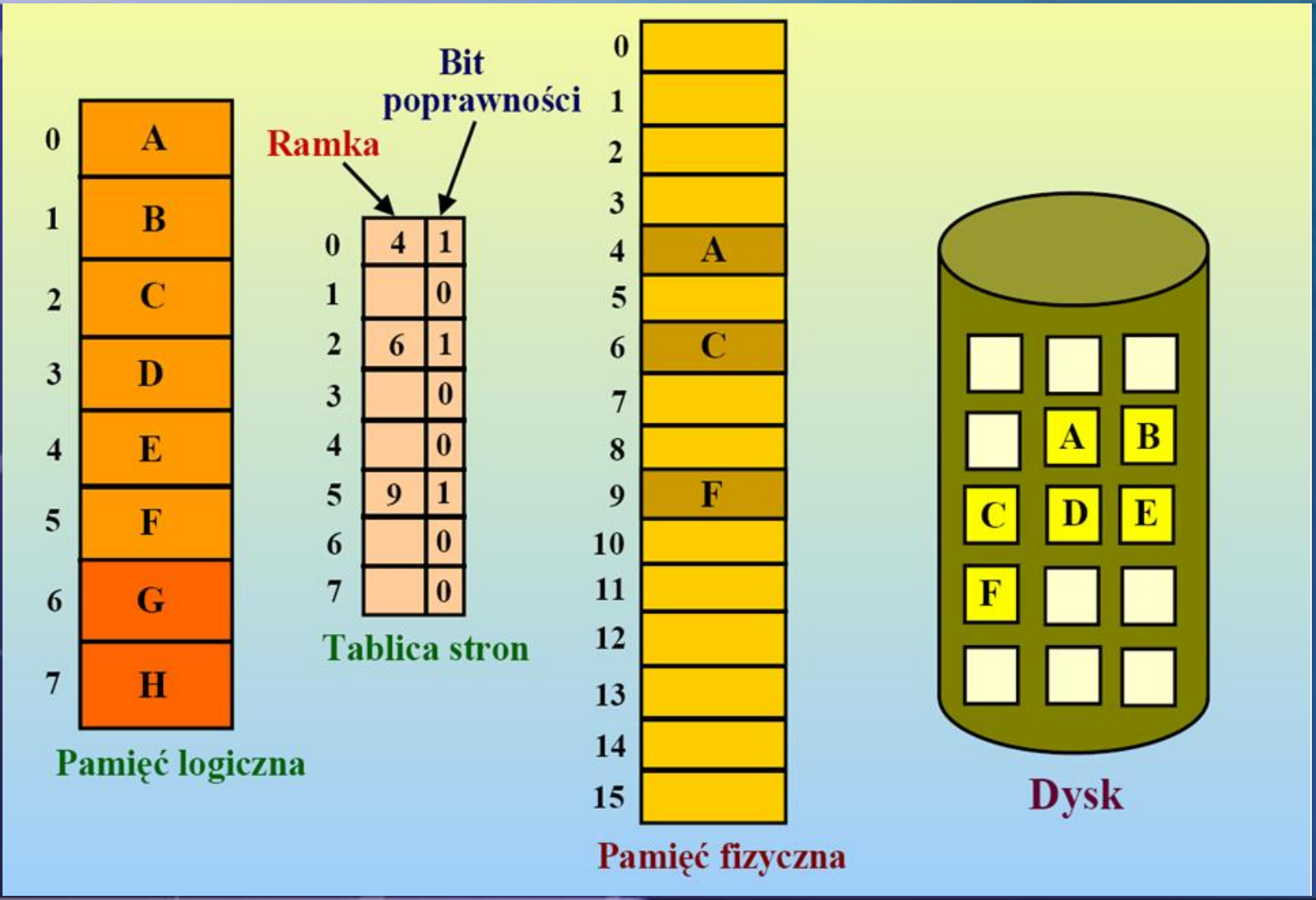
Stronicowanie pamięci

- Eliminuje się fragmentację zewnętrzną, ale pozostaje fragmentacja wewnętrzna.
- Adres logiczny generowany przez proces składa się z dwóch części:
  - numeru strony s *(page numer)*– indeks w tablicy stron (tablica stron zawiera adresy bazowe wszystkich stron w pamięci operacyjnej).
  - odległość na stronie o *(page offset)* – w połączeniu z adresem bazowym daje adres fizyczny pamięci, który jest wysyłany do jednostki pamięci.

Odwzorowanie adresu



Tablica stron z brakami stron w pamięci



Zarządzanie pamięcią

- Domyślny rozmiar procesu w 32-bitowym systemie Windows wynosi 2 GB.
- Jeśli plik wykonywalny jest oznaczony jako wymagający dużej przestrzeni adresowej proces 32-bitowy może osiągnąć 3 GB w systemie 32-bitowym lub 4 GB w systemie 64-bitowym.
- Wirtualna przestrzeń adresowa procesu w 64-bitowym systemie Windows od wersji 8.1 wynosi 128 TB.
- Maksymalny rozmiar pamięci fizycznej obsługiwanej przez system Windows waha się od 2 GB do 24 TB w zależności od wersji i edycji systemu.

Zadania Menedżera pamięci

- Odwzorować wirtualną przestrzeń adresową procesu na pamięć fizyczną,
- Przeprowadzić stronicowanie części zawartości pamięci na dysk w momencie, gdy działające wątki starają się użyć więcej pamięci fizycznej, niż jest aktualnie jest dostępne,
- Ponownie przenieść tę zawartość z dysku do pamięci fizycznej, gdy jest potrzebna.

Komponenty Menedżera pamięci

- Menedżer pamięci systemu Windows - część centrum wykonawczego.
- Zestaw systemowych usług wykonawczych.
- Obsługa wykrytych sprzętowo wyjątków w zarządzaniu pamięcią.
- Procedury działające jako wątki trybu jądra w ramach procesu *System*.
  - Menedżer zestawu równowagi
  - Procedura wymiany procesu/stosu
  - Moduł zapisu stron zmodyfikowanych
  - Moduł zapisu stron zmapowanych
  - Wątek dereferencji segmentu
  - Wątek zerowania stron

Strony małe i duże

- Strona jest najmniejszą jednostką chronioną na poziomie sprzętowym.
- Procesory współpracujące z systemem Windows obsługują strony dwóch rozmiarów:
  - małe
  - duże.
- Rzeczywiste rozmiary stron zależą od architektury procesora.

Architektura	Rozmiar małej strony	Rozmiar dużej strony	Liczba małych stron w dużej stronie
x86 (PAE)	4 kB	2 MB	512
x64	4 kB	2 MB	512
ARM	4 kB	4 MB	1024