11. STRUKTURA DYSKU

Napęd dyskowy adresowany jest jak jednowymiarowa tablica *bloków logicznych*, które sekwencyjnie odwzorowywane są na fizyczne sektory dysku.

Czas szukania *(seek time)* -czas przesunięcia ramienia z głowicami na zadany sektor.

Opóźnienie obrotowe *(rotational latency)* -czas obrotu dysku do wybrania zadanego sektora.

Szerokość pasma *(bandwidth)* -liczba przesłanych **bajtów**, podzielona przez czas zawarty między 1-szym zamówieniem usługi dyskowej a zakończeniem ostatniego przesłania.

➔ Planowanie operacji dyskowych optymalizuje: -**czas** dostępu,
-**szerokość** pasma.

➔ Proces składa zamówienie do SO na wykonanie dyskowej operacji We/Wy.

Zamówienie zawiera parametry przesyłania: -rodzaj operacji We/Wy,
-adres w PAO,
-adres dyskowy,
-liczbę bajtów.

Jeśli napęd/sterownik są zajęte, to nowe zamówienie ustawione jest w **kolejce** zamówień.
SO dokonuje **wyboru** zamówienia, które zostanie obsłużone w **następnej** kolejności.

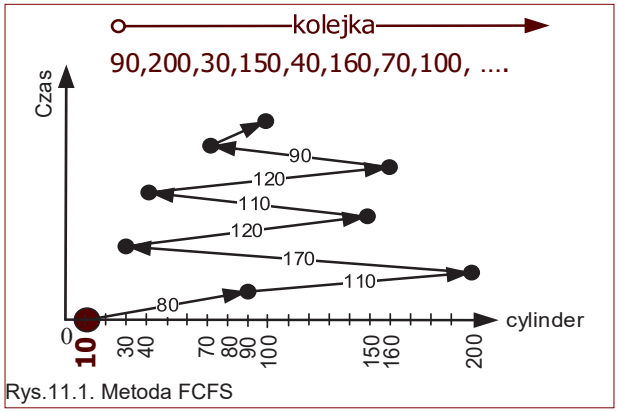
❒ **Algorytm planowania** dostępu **FCFS**

Algorytm „pierwszy zgłoszony - pierwszy obsłużony" (***F**irst **C**ome, **F**irst **S**erved*).

Dyskowa **kolejka** zamówień dotyczy bloków w cylindrach:

90, 200, 30, 150, 40, 160, 70, 100, ...

W chwili t_0 głowica dysku znajduje się w cylindrze **10**.



Rys.11.1. Metoda FCFS

Najpierw przemieści od cylindra **10** do cylindra **90**, a następnie do cylindra 200, 30, 150, itd.
Dokonano łącznego przejścia 830 cylindrów.

Wada algorytmu: gwałtowne ruchy głowicy (np. od cylindra 150 do 40 i z powrotem do 160).

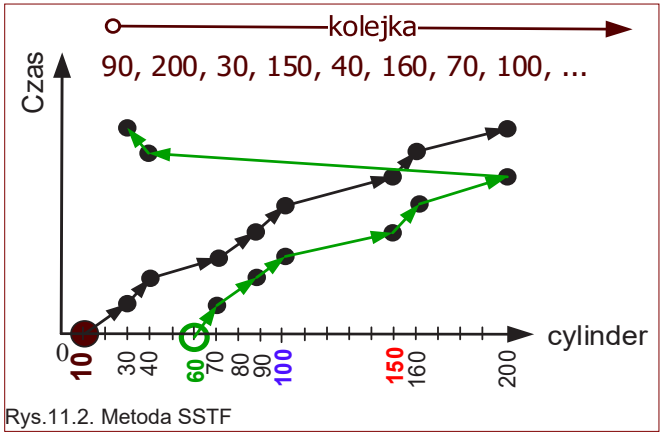
Gdyby zamówienia odnoszące się do cylindrów **30** i **40** były obsłużone razem, przed/po zamówieniach na cylindry 150 i 160, to ruch głowicy zmalałby, polepszając wydajność.

❒ **Algorytm planowania** dostępu **SSTF** (***S**hortest **S**eek **T**ime **F**irst* **SSTF**).

Obsługiwane są zamówienia sąsiadujące z bieżącą pozycją głowicy, zanim nastąpi jej przemieszczenie w dalsze rejony.

W algorytmie **SSTF** „**najpierw najkrótszy czas przeszukiwania**" wybiera się zamówienie z **najmniejszym** czasem przeszukiwania względem ***bieżącej pozycji*** głowicy.

Najbliższe zamówienie względem początkowego położenia głowicy (**10**) dotyczy cylindra **30**.
Od cylindra (**30**) najbliższe zamówienie dotyczy cylindra **40**.
Od cylindra (**40**) najbliższe zamówienie dotyczy cylindra **70**.



Rys.11.2. Metoda SSTF

Metoda wymaga łącznego przejścia **190** cylindrów, czyli ponad **cztery** razy mniej niż metoda **FCFS**.

Czas przeszukiwania wzrasta proporcjonalnie do liczby cylindrów odwiedzanych przez głowicę, dlatego wybiera się zamówienie **najbliższe bieżącemu położeniu** głowicy.

Algorytm **SSTF** znaczne polepsza wydajności.

◗ **Algorytm SSTF nie jest idealnie optymalny.**

W chwili t_0 głowica znajduje w cylindrze **60**, to najpierw przemieści się cylindra **70**, następnie kolejno: **90 → 100 → 150 → 160 → 200 → 40 → 30**.

◗ **Planowanie metodą SSTF może powodować *głodzenie* pewnych zamówień.**

☛ Zamówienia mogą nadchodzić w dowolnych chwilach.

W kolejce są dwa zamówienia odnoszące się do cylindrów: **100**, **150**.

Jeśli podczas obsługi zamówienia w cylindrze **100** nadejdzie zamówienie ***bliskie*** cylindra **100**, to zostanie ono obsłużone, jako następne, powodując dalsze oczekiwanie zamówienia do cylindra **150**.

☛ Ciąg zamówień bliskich cylindra **100**, spowoduje ***nieskończone*** **oczekiwanie** do cylindra **150**.

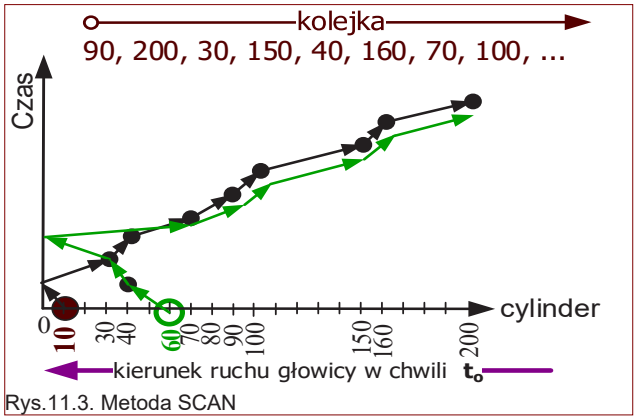
❒ **Algorytm planowania dostępu SCAN** *(omiatanie)*

Głowica nieprzerwanie przeszukuje cały dysk tam i z powrotem.

Głowica rozpoczyna ruch od jednej krawędzi i przemieszcza się w kierunku przeciwległej, obsługując wszystkie zamówienia po drodze.

Gdy dotrze do skrajnego cylindra ➔ zmienia się kierunek ruchu głowicy i obsługa jest kontynuowana.

SCAN to **algorytm windy** *(elevator algorithm)*; ramię dysku zachowuje się jak winda, obsługując najpierw zamówienia w kierunku do góry, później na dół.



Rys.11.3. Metoda SCAN

W chwili t_0 głowica dysku znajduje się w cylindrze **(10)** i przemieszcza się w kierunku cylindra **0**.

Przy cylindrze **0** ramię zmienia kierunek ruchu i przemieszcza się w kierunku przeciwnym, obsługując zamówienia w cylindrach: 30, 40, 70, 90, 100, 150, 160, 200

- Jeśli w chwili t_0^+ w kolejce pojawi się zamówienie do cylindra **8**, to natychmiast zostanie zrealizowane.

W chwili t_0 głowica dysku znajduje się w cylindrze **(60)** i przemieszcza się w kierunku cylindra **0**.

Obsługiwane są zamówienia w cylindrach **30**, **40**, następnie po zmianie kierunku: 70, 90, 100, 150, 160, 200.

❒ **Algorytm planowania dostępu C-SCAN** (*circular SCAN* - omiatanie cykliczne)

Głowica przesuwa się od krawędzi do środka, obsługując napotykane zamówienia. Gdy osiągnie skrajne położenie, wraca natychmiast do krawędzi zewnętrznej, nie obsługując zamówień w drodze powrotnej.

Cylindry traktowane są jak lista cykliczna, w której ostatni cylinder spotyka się z pierwszym.

❒ **Algorytm planowania dostępu LOOK** i **C-LOOK**

W algorytmach SCAN i C-SCAN głowica przemieszcza się między **skrajnymi położeniami**.

Głowica przesuwa się między skrajnymi zamówieniami w każdym kierunku i nie dochodzi do skrajnego położenia na dysku.

Algorytmy o takim działaniu nazywają się LOOK i C-LOOK, ponieważ przed kontynuowaniem ruchu, „**patrzy się**", czy w danym kierunku znajduje się jeszcze zamówienie.

◗ **Jak wybrać najlepszy algorytm planowania dostępu do dysku?**

Planowanie metodą **SSTF** jest dość powszechne i wygląda naturalnie.

Algorytmy **SCAN** i **C-SCAN** są odpowiedniejsze przy dużej liczbie zamówień na operacje dyskowe, gdyż mniejsze jest prawdopodobieństwo występowania problemu głodzenia.

Gdy kolejka ma jedno nie obsłużone zamówienie, to wszystkie algorytmy planowania dadzą ten sam rezultat, co metoda **FCFS**.

- Dla **ciągłej** metody przydzielania bloków, program czytający wygeneruje kilka zamówień odnoszących się do sąsiednich miejsc na dysku, co ogranicza ruch głowicy.
- Plik **listowy**/**indeksowy** zawiera bloki rozrzucone po dysku, powodując większy ruch głowic.

➔ Ważna jest lokalizacja katalogów i bloków indeksowych.

Każdy plik otwierany jest przed użyciem, co wymaga przeszukania struktury katalogowej.

Gdy wpis katalogowy znajduje się w **1-szym** cylindrze, a **dane** pliku w **ostatnim** to głowica musi przebyć całą szerokość dysku.

Algorytm planowania dostępu do dysku powinien być napisany jako osobny moduł systemu operacyjnego, aby można go było łatwo konserwować.

◗ Tylko **odległość** szukania brana jest pod uwagę w przedstawionych algorytmach

Opóźnienie obrotowe może być również duże jak średni czas szukania.

Uwzględnianie parametru opóźnienia obrotowego napotyka w SO na trudności, ponieważ dyski nie ujawniają fizycznego położenia bloków logicznych.

- Algorytmy planowania dostępu do dysku często implementowane są w sterowniku ***wbudowanym*** w napęd dysku.

◗ **Czy system operacyjny może przerzucić planowanie dostępu do dysku na sprzęt ?**

SO **powinien** uwzględniać różne ograniczenia na porządek obsługiwania zamówień.

- **Stronicowanie** na żądanie wymaga **pierwszeństwa** nad operacjami We/Wy **aplikacji**.
- Operacje **Pisania** są pilniejsze niż **C**zytania, jeżeli w pamięci **cache** ***brakuje*** wolnych stron.

System operacyjny musi realizować własne algorytmy planowania dostępu do dysku i sukcesywnie dostarczać zamówienia sterownikowi dysku.

11.1. Zarządzanie obszarem wymiany

Jest to zadaniem systemu operacyjnego.

Pamięć wirtualna korzysta z przestrzeni dyskowej jak z rozszerzenia pamięci głównej.

Dla obszaru wymiany istotna jest najlepsza **przepustowość** pamięci wirtualnej.

Bezpieczniej jest nadmiernie oszacować wielkość obszaru wymiany niż niedoszacować, gdy systemowi zabraknie obszaru wymiany, to może **zaniechać** wykonania procesów.

Obszar wymiany może być**:** -utworzony ze zwykłego systemu plików,
-znajdować w osobnej strefie dyskowej.

Jeżeli jest plikiem w obrębie systemu plików, to do jego utworzenia i przydzielenia mu miejsca można użyć zwykłych procedur systemu plików.

Podejście łatwe w realizacji, lecz mało wydajne, gdyż poruszanie się w strukturze katalogowej i strukturach danych związanych z przydzielaniem miejsca na dysku jest czasochłonne.

Fragmentacja zewnętrzna może wydłużyć czas wymiany, podczas wielokrotnych przeszukiwań w czasie **C**zytania lub **Z**apisu obrazu procesu.

◗ Obszar wymiany częściej tworzony jest w osobnej strefie dyskowej

Nie lokalizuje się w niej żadnego systemu plików ani **nie** buduje struktury katalogowej.

Do przydzielania i zwalniania bloków stosuje się ***ZARZĄDCĘ pamięci obszaru wymiany***.

☛ Optymalizuje się tylko **szybkość** wymiany a nie zużycie pamięci.

Wewnętrzna fragmentacja jest akceptowalna, gdyż:
-dane w obszarze wymiany pozostają krótko,
-duża częstotliwość dostępu do obszaru wymiany.

W UNIX implementowanie wymiany rozpoczęto od kopiowania całych procesów z PAO do ciągłych obszarów dysku i z powrotem.

Udostępnienie sprzętu stronicującego zaowocowało implementacją **kombinacji wymiany** i **stronicowania**.

Niezawodność dysku

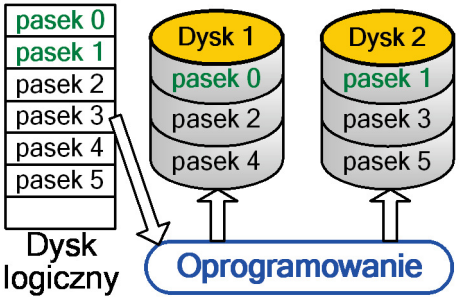
Dyski są najbardziej zawodną częścią systemu.

Nadmiarowa tablica niezależnych dysków (***R**edundant **A**rray of **I**ndependent **D**isks* - **RAID**).

Metoda **paskowania dysku** *(disk striping),* traktuje grupę dysków jak jedną jednostkę pamięci.

Wszystkie dane znajdują się na jednym dysku logicznym.

Dysk dzielony jest na **paski** (fizyczne bloki, sektory, inne jednostki), które odwzorowywane są cyklicznie na kolejne elementy macierzy.



Pasmo to zbiór logicznie ciągłych pasków
(jeden pasek na jeden fizyczny element macierzy).

W macierzy **n-**dyskowej pierwszych **n** logicznych pasków przechowuje się w pierwszych paskach na każdym z **n** dysków fizycznych.
(tworzone jest pierwsze pasmo.)

Jeśli jedno żądanie We/Wy dotyczy wielu logicznie ciągłych pasków, to **n** pasków można obsłużyć równolegle.

RAID 0 rozprasza dane użytkownika i systemu po wszystkich dyskach.

Jeżeli zgłoszono **dwa** żądania dostępu do **2-ch** różnych bloków danych to jest prawdopodobne, że bloki te znajdują się na różnych dyskach i żądania będą zrealizowane równolegle.

W środowisku transakcyjnym może występować duża liczba żądań We/Wy na sekundę.
Macierz dyskowa zwiększa tempo ich realizacji, rozkładając obciążenie na wiele dysków.

Jeśli paski są tak **duże**, że jedno żądanie We/Wy wymaga tylko jednego dostępu do dysku, to można równolegle obsłużyć wiele oczekujących żądań We/Wy,

Zsynchronizowanie obrotów dysków polepsza wydajność, gdyż wszystkie dyski są gotowe do przesyłania podbloków w tym samym czasie.

Schematy RAID mogą polepszać niezawodność, pamiętając nadmiarowe dane.

Odbicie lustrzane *(mirroring)* lub **tworzeniem cienia** (*shadowing),* utrzymuje kopię każdego dysku.

RAID 1 zapewnia nadmiarowość poprzez **powielanie** danych.

Dane dzielone są na paski, i każdy pasek logiczny jest odwzorowany na **dwa odrębne** dyski fizyczne.

☛ Każdy dysk w macierzy ma dysk lustrzany, zawierający te **same dane**.

Kiedy dysk ulegnie awarii, dane można odczytać z drugiego dysku.

RAID 3 to striping na poziomie **bajtów**, z zapisem kodów kontrolno-redundancyjnych i parzystości na wydzielonym dysku;

Zapewnia odtworzenie danych po uszkodzeniu dysku.

RAID 4 to striping **bloków** danych z zapisem kodów kontrolno-redundancyjnych na wydzielonym dysku;

Zapis kodów kontrolnych na osobnym dysku dla bloków danych stanowi „wąskie gardło" systemu, przydatność dla systemów transakcyjnych.

Przeplatanie bloków parzystości *(block interleaved parity)*:

mała część obszaru dysku jest wykorzystywany do przechowywania bloków parzystości.

RAID 5 to striping **bloków** danych z **rozproszonym** zapisem kodów kontrolno-redundancyjnych;

Zapewnia wysoki poziom bezpieczeństwa danych przy najniższym koszcie.

Idea pracy RAID 5 jest podobna do RAID 3.

Dane o parzystości są równomiernie rozłożone na wszystkich dyskach macierzy; **nie są** składowane na jednym wydzielonym dysku.

RAID 10 (kombinacja poziomu 0/1) to striping danych na lustrzanych dyskach;

Efektywna szybkość i bezpieczeństwo; duży koszt gdyż 50% pojemności dysków jest niedostępnych dla użytkownika (obszar na redundancję danych).

Zastosowanie do **rekonstrukcji** po **awariach nadmiarowych danych na dyskach RAID sprawia, że utrata danych występuje raz na wiele lat.**

Anex 11.1

W UNIX 4.3BSD **uruchamianemu P**rocesowi **przydziela** się obszar wymiany.

Na dysku rezerwuje się obszar na:

-**program**, zwany też ***stronami tekstu***

-**segment danych** procesu.

Wstępny przydział całego obszaru gwarantuje procesowi ciągły dostęp do obszaru wymiany.

Rozpoczynając proces SO sprowadza z plików dyskowych **strony** jego programu, które ***mogą być*** wysłane do obszaru wymiany i z tego obszaru czytane w dalszej działalności procesu.

Taka organizacja ułatwia korzystanie przez kilka procesów z tych samych stron programu.

Kontakt z systemem plików może przypadać tylko ***po razie*** na każdą stronę programu.

Strony **segmentu danych** czytane są z systemu plików (lub tworzone) i **zostają zapisane** w obszarze wymiany, skąd sprowadzane w miarę potrzeb.

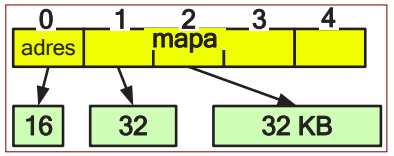
Jądro korzysta z dwu procesowych **map wymiany** *(swap maps)*.

Strony tekstu mają ustalony rozmiar, obszar wymiany przydziela się blokami po **512** KB, z wyjątkiem ostatniego kawałka programu, przydzielanego, co **1KB**. 512|512|512|21

Dla **segmentu danych** mapa wymiany ma ustalony rozmiar,

ale adresy wskazują na ***bloki wymiany*** o **zmiennych** rozmiarach, gdyż segment danych może rosnąć z upływem czasu.

Blok wskazywany z **k**-tej pozycji mapy wymiany ma rozmiar $\mathbf{2^k}$ x **16** KB, jego maksymalna wielkość to **2** MB.



Jeśli proces zwiększył **segment danych** poza ostatnio przydzielony blok w obszarze wymiany, to SO przydziela mu nowy blok, **dwa razy** większy niż poprzedni.

Małe procesy używają małych bloków, co zmniejsza również fragmentację.

SunOS4 podczas wykonywania procesu strony programu sprowadzane są z plików do PAO, gdzie są udostępniane **(usuwane gdy są zbędne**).

Powtórne **czytanie strony z pliku** jest wydajniejsze niż zapisywanie jej w obszarze wymiany i czytanie jej stamtąd.

Solaris 2 przydziela obszar wymiany wówczas, gdy strona jest wyrzucana z pamięci fizycznej, a **nie podczas pierwszego** utworzenia strony pamięci wirtualnej.

Wpływa to na wzrost wydajności komputerów - mających dużo PAO – poprzez mniejszą intensywność stronicowania.