9. Řízení vnější paměti

01 K čemu slouží vnější paměť

02 Charakteristika HDD

03 Metody přidělování místa na disku

- a) Spojité
- b) Spojitý seznam
- c) Indexová alokace

04 Plánovací metody přístupu na disk

- a) FCFS
- b) SSTF
- c) SCAN
- d) C-SCAN
- e) LOOK
- f) C-LOOK

06 HDD vs. SSD

- a) Defregmentace

9. Řízení vnější paměti

Účel vnější paměti

Vnější paměť *(angl. secondary storage)* slouží k **trvalému uchovávání dat a programů i po vypnutí počítače**. Na rozdíl od operační paměti *(RAM)*, jejíž obsah se při vypnutí ztrácí, vnější paměti jsou **energeticky nezávislé – data v nich zůstávají uložena trvale**.

Charakteristické rysy:

- **Procesor nemá** přímý **přístup** k **vnější paměti** *(např. disku)*. Přístup **zajišťuje operační systém pomocí ovladačů zařízení** *(device drivers)*, které komunikují s řadičem paměti *(např. SATA nebo NVMe kontrolér)*.
- Data jsou na discích **organizována do souborů**, a to **pomocí souborového systému** *(např. NTFS, FAT32, exFAT, ext4 atd.)*.

Výhody vnější paměti:

- Nízké náklady na GB *(zejména u HDD)*.
- **Energetická nezávislost** – obsah se neztratí po vypnutí.
- **Nedestruktivní čtení** – čtení dat nijak nepoškozuje jejich obsah, což je rozdíl oproti některým starším typům pamětí.

Typy vnější paměti:

- **Stálé** *(pevně vestavěné):*
  - **HDD** *(pevný disk)*
  - **SSD** *(polovodičové úložiště)*
- **Výměnné** *(externí nebo přenosné):*
  - **Diskety** *(historicky)*
  - **CD/DVD** *(optická média)*
  - **USB flash disky** *(moderní výměnná paměť)*

Charakteristika pevného disku (HDD – Hard Disk Drive)

Pevný disk slouží k **trvalému nebo dočasnému ukládání dat** – např. **programy, uživatelská data i samotný operační systém**. Uchovávání informací probíhá pomocí **magnetické indukce.**

Zápis a čtení dat:

- Záznam probíhá **změnou magnetické orientace částic na povrchu plotny** *(reprezentace bitů 1 a 0)*.
- Magnetická hlava **vytváří změnou polarity napětí v cívce magnetické pole**, které mění **orientaci částic ve vrstvě záznamu**.
- Pro **čtení** se používá **MR hlava** *(magnetorezistivní)* – ta **měří změny odporu v závislosti na magnetickém poli** záznamové vrstvy.

Konstrukce disku:

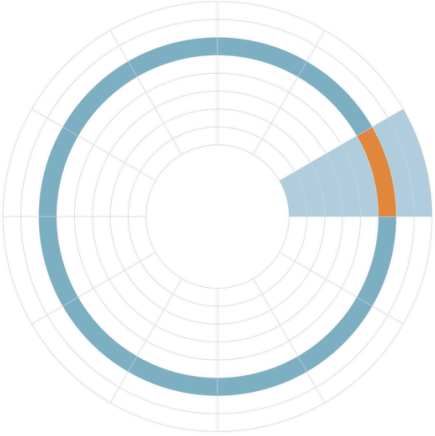
- Samotné **plotny disku** jsou z **nemagnetického materiálu** *(např. hliník nebo sklo)*.
- **Povrch** je **pokryt feromagnetickou vrstvou**, často z **oxidu železa** *(Fe_2O_3)*, která umožňuje magnetický zápis.

9. Řízení vnější paměti

Geometrie disku a organizace dat:

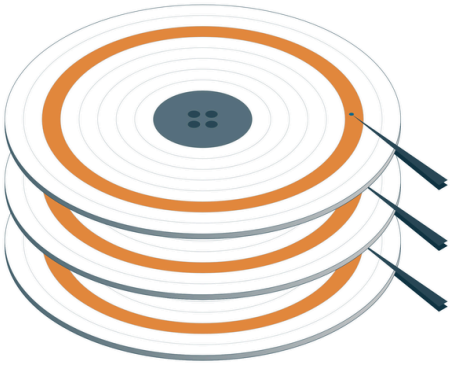
Stopa *(Track)*:

- Je to **soustředná kružnice na povrchu každé plotny**.
- Slouží k **záznamu dat**.
- Číslování stop začíná od **0** *(vnější okraj)* směrem **ke středu** disku.



Sektor *(Sector)*:

- Je to **výseč jedné stopy** – **nejmenší fyzická jednotka**, do které lze data zapisovat nebo je číst.
- Obvykle má velikost **512 bajtů** *(u moderních disků i 4 kB – tzv. Advanced Format)*.
- **Obsahuje:**
  - **Identifikační část** – adresa sektoru ve formátu CHS (Cylinder, Head, Sector)
  - **Datovou část** – např. 512 B
  - **CRC kód** – pro detekci a opravu chyb



Cylindr *(Cylinder)*:

- **Skládá se ze stop** se stejným číslem na všech plotnách disku, které **leží nad sebou v zákrytu.**
- Umožňuje, aby všechny hlavy současně četly nebo zapisovaly data na stejné pozici každé plotny.
- Číslování cylindrů rovněž od vnějšího okraje dovnitř.

Dá se zmínit i více (viz otázka o HDD discích), ale nedoporučuju protahovat a kecat jen o tom! Musíte se u zkoušky dostat k nejlépe všem podotázkám!

Metody přidělování místa na disku – spojité přidělování (souvislá alokace)

Základní charakteristika:

Spojité přidělování *(contiguous allocation)* je **nejjednodušší a historicky nejstarší způsob správy místa** na disku.
Každý soubor je **uložen v souvislém** *(spojitém)* **bloku diskového prostoru** – tedy v **blocích, které na sebe přímo navazují**.

- **Uživatel/OS zná začátek souboru** a jeho **délku**, což umožňuje efektivní čtení dat.
- Výhodou je **rychlý a snadný přístup**, **nízká fragmentace** dat při zápisu, ale má **omezení v pružnosti práce** s většími nebo dynamicky rostoucími soubory.

Výhody:

- **Přímý i sekvenční přístup** k datům *(vysvětleno níže)*.
- **Malý pohyb hlaviček disku** – protože data jsou fyzicky blízko sebe.
- **Rychlé čtení** díky předvídatelné struktuře.

9. Řízení vnější paměti

Nevýhody:

- Pokud se soubor **zvětší**, je nutné ho **přesunout do většího souvislého bloku**, což je **neefektivní**.
- Velké soubory často **není kam uložit**, i když je na disku celkově dost volného místa → **vzniká fragmentace** volného místa.
- **Nutnost defragmentace disku**, aby vznikl nový souvislý prostor.
- Těžko se předem odhaduje, kolik místa bude soubor potřebovat.

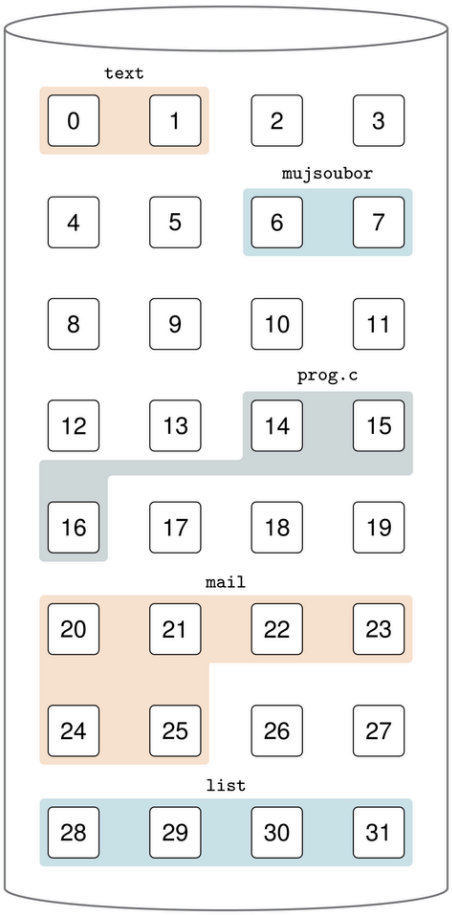
Typy přístupu k datům:

- **Sekvenční přístup:**
  - Data se **čtou postupně**, od začátku souboru až po požadovanou informaci.
  - Nutné **přečíst všechny předchozí bloky**.
  - Vhodné pro média jako **pásky nebo streamované čtení**.
- **Přímý** *(náhodný)* **přístup:**
  - Lze **okamžitě přejít na požadovaný blok** díky **znalosti počáteční adresy** a **velikosti**.
  - Typický pro pevné disky a SSD.

Alokační strategie *(algoritmy pro výběr volného místa):*

Když systém hledá vhodné místo pro nový soubor, využívá jeden z těchto algoritmů:

- **First Fit** *(první vhodné místo):*
  - Vybere **první volný blok**, do kterého se soubor vejde.
  - Nejjednodušší a nejčastěji používaný, rychlá implementace.
- **Best Fit** *(nejvhodnější místo):*
  - Vybere **nejmenší možný blok**, který soubor pojme → minimalizuje nevyužitý prostor.
  - Může vést ke vzniku mnoha malých mezer (externí fragmentace).
- **Last Fit** *(poslední vhodné místo):*
  - Vybere **poslední nalezený blok**, do kterého se soubor vejde.
  - Méně běžný, někdy vhodný pro specifické optimalizace.
- **Worst Fit** *(největší vhodné místo):*
  - Vybere **největší dostupný blok**, do kterého se soubor vejde.
  - Cílem je zachovat větší menší bloky pro budoucí soubory → strategie funguje opačně než Best Fit.



Adresář

Soubor	Start	Počet
text	0	2
mujsoubor	6	2
prog.c	14	3
mail	20	6
list	28	4

9. Řízení vnější paměti

Metody přidělování místa na disku – spojitá alokace (seznam / řetězcová alokace)

- Tato metoda **nevyžaduje souvislý prostor na disku** – **soubor může být rozdělen** na více **nesouvislých bloků**, které jsou mezi sebou **logicky propojeny ukazateli** *(adresami)*.
- Pro správu **postačí znát počáteční blok** souboru a **informaci o konci souboru** *(EOF)*.
- Používá se například v **souborových systémech FAT12, FAT16 a FAT32**, známých z **MS-DOS a Windows 95/98.**

Vlastnosti a výhody:

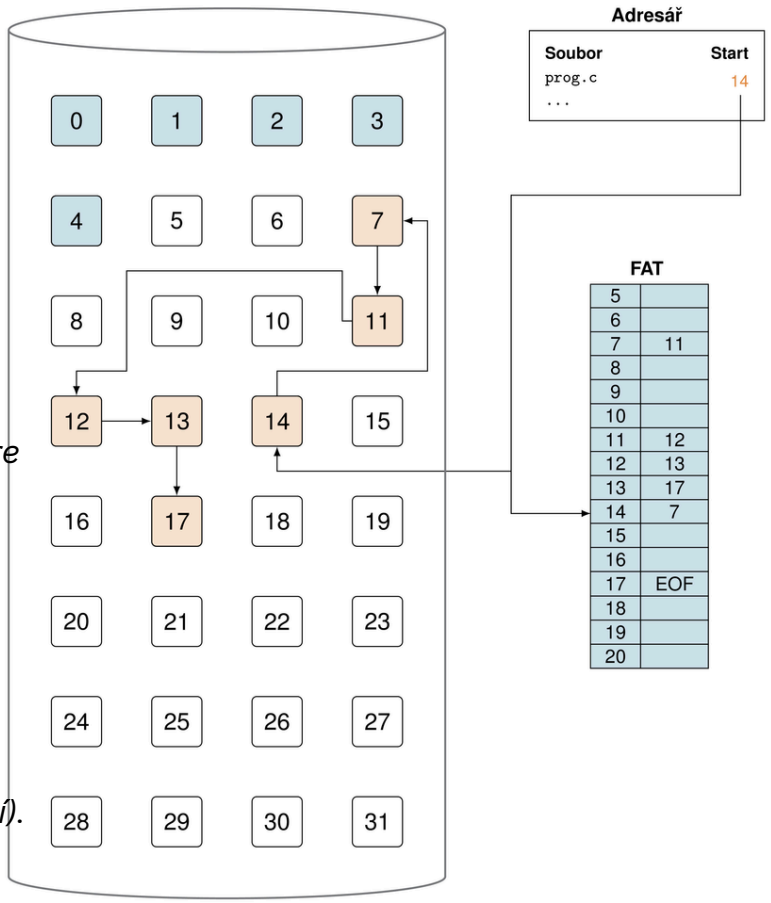
- **Není nutné souvislé místo** – eliminuje problém s fragmentací volného místa *(vnější fragmentace)*.
- **Bloky se alokují až v okamžiku zápisu** – žádné předem rezervované volné bloky → úspora místa.
- **Bloky se ukládají co nejblíže** – minimalizuje se pohyb hlaviček disku a zrychluje čtení.

Způsob přístupu:

- **Sekvenční přístup** – základní způsob.
- Bloky jsou **navzájem propojené ukazateli** a přístup k určitému místu vyžaduje **procházení všech předchozích bloků**.
- **Přímý přístup** *(až při použití FAT tabulky)*:
  - **FAT** *(File Allocation Table)* je tabulka, která **mapuje soubor na jednotlivé bloky disku**.
  - Každý **záznam v tabulce obsahuje odkaz na následující blok**.
  - **Poslední blok** obsahuje značku **EOF** *(End of File)*.
  - Díky tomu lze **rychle vyhledat konkrétní blok**, i když fyzicky nejsou po sobě.

Alokační jednotky:

- **Cluster** *(alokovací blok)* je **nejmenší jednotka**, která se **přiděluje souboru**.
- Velikost clusteru závisí na **souborovém systému a kapacitě disku** *(např. u FAT32 může být 4 kB, ale i více)*.



Fragmentace:

- **Vnitřní fragmentace:**
  - Vzniká, pokud **soubor nevyužije celý poslední alokační blok** *(např. soubor má 6,1 kB, ale bloky jsou po 4 kB → zabere 2 bloky = 8 kB → 1,9 kB je nevyužito)*.
- **Vnější fragmentace je potlačena**, protože soubory nemusí být uložené v souvislém prostoru.

Řetězení bloků – bez a s FAT tabulkou:

- **Bez FAT tabulky:**
  - Každý blok **obsahuje adresu dalšího bloku přímo v sobě** *(tzv. fyzické řetězení)*.

- **S FAT tabulkou:**
  - FAT obsahuje pro **každý blok index dalšího bloku**.
- **Například:**
  - Blok 5 ➜ 12, Blok 12 ➜ 24, Blok 24 ➜ EOF
- Díky tomu se oddělují **data od metadat**, čímž je **přístup rychlejší a správa efektivnější**.

Metody přidělování místa na disku – indexová alokace

Indexová alokace *(indexed allocation)* **ukládá všechny adresy** *(indexy)* **datových bloků daného souboru do speciálního indexového bloku.**

- Každý **soubor** má vlastní **indexový blok**.
- Indexový blok **obsahuje seznam adres bloků**, které obsahují **skutečná data souboru**.
- Zpočátku jsou položky **indexu nastaveny na -1** *(neobsazeno)*, později se **nahradí čísly skutečných bloků** podle pořadí.

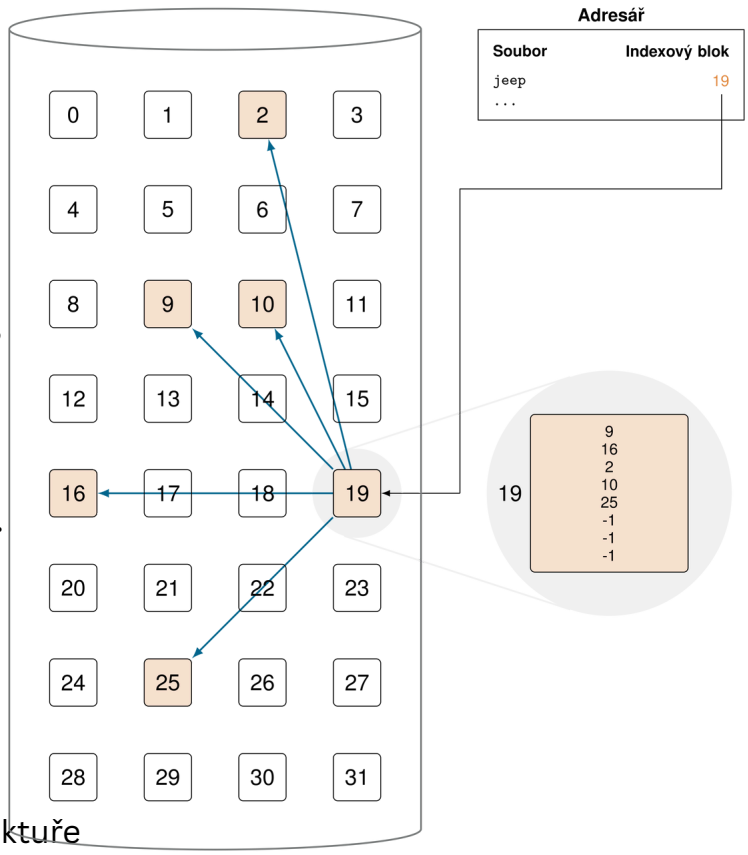
Vlastnosti a výhody:

- **Přímý i sekvenční přístup** – díky indexům lze přejít na libovolný blok i číst data postupně.
- **Potlačená vnější fragmentace** – bloky souboru nemusí být fyzicky vedle sebe.
- Přístup k souboru je **rychlý**, protože indexový blok je při práci načten do operační paměti *(RAM)*.
- **Lze snadno spravovat dynamicky rostoucí soubory** – jen se přidají další položky do indexu.

Nevýhody:

- **Vnitřní fragmentace** – poslední datový blok často není zcela využit.
- **Složitější implementace** než u spojité nebo řetězcové alokace.
- Složitost roste u velkých souborů, které vyžadují více indexových struktur.

Optimalizace a struktury indexace:



Spojová struktura:

- Používá **jeden nebo více indexových bloků** za sebou. Např:
  - Velikost bloku = **2048 B**
  - **Jeden ukazatel zabírá 4 B ➜ vejde se 512** ukazatelů
  - Každý **ukazatel adresuje 1 blok** *(např. 2048 B)* ➜ pokryjeme **512 × 2048 B = 1 MB dat**

Víceúrovňový index:

- Hlavní indexový blok **neukazuje přímo na datové bloky**, ale na **druhotné indexové bloky.**
- Ty pak obsahují odkazy na **skutečná data ➜ hierarchická struktura**.
- **Výhoda:** zvládne i velmi velké soubory *(např. gigabajtové či víc)*.

Kombinovaný přístup:

- Používá více způsobů podle velikosti souboru:
  - **Malé soubory** – ukládají data přímo ve struktuře *(např. přímo v i-nodu)*, bez indexového bloku.
  - **Větší soubory** – využívají jednoúrovňový nebo víceúrovňový index.

9. Řízení vnější paměti

Další informace:

- Využíváno např. v **souborových systémech UNIX/Linux pomocí i-nodů** *(index nodes)*.
- Snaha mít bloky souboru **co nejblíže u sebe kvůli minimalizaci pohybu hlaviček disku při čtení**.
- Také je **snaha o co nejmenší indexový blok**, aby nezabíral zbytečně paměť.

Plánovací metody přístupu na disk

Při přístupu na disk je **nutné naplánovat, který požadavek bude obsloužen jako první**. Cílem plánování je minimalizovat dobu čekání a zkrátit celkový čas potřebný k obsluze požadavků na čtení/zápis.

Přístup na disk se skládá ze 3 částí:

1. **SEEK** *(vyhledání stopy / cylindru):*
   - Pohyb **čtecí/zapisovací hlavy** nad požadovanou stopu.
   - Největší **zdroj zpoždění** *(hlavičky se musí fyzicky přesunout)*.
   - Rychlost závisí na vzdálenosti mezi aktuální a cílovou stopou.
2. **LATENCY** *(otáčková latence):*
   - **Čekání**, než se požadovaný sektor disku natočí pod hlavičku.
   - Závisí na otáčkách disku *(RPM)* – např. u **7200 ot./min** je průměrná latence asi **4,17 ms**.
3. **TRANSFER** *(přenos dat):*
   - Přenesení dat **z disku do paměti** *(čtení)* nebo **na disk** *(zápis)*.
   - **Trvá nejméně času** v porovnání s předešlými dvěma kroky.

Plánovací metoda: FCFS (First Come, First Serve)

- *"Kdo dřív přijde, ten dřív mele."*
- Diskové požadavky **jsou obsluhovány v pořadí, v jakém přicházejí** – bez ohledu na pozici hlavičky.
- Každý požadavek **je zařazen do fronty a odbavován postupně**.

Vlastnosti:

- **Jednoduchá implementace** – snadno se programuje.
- **Spravedlivé pořadí** – žádný požadavek není preferován.

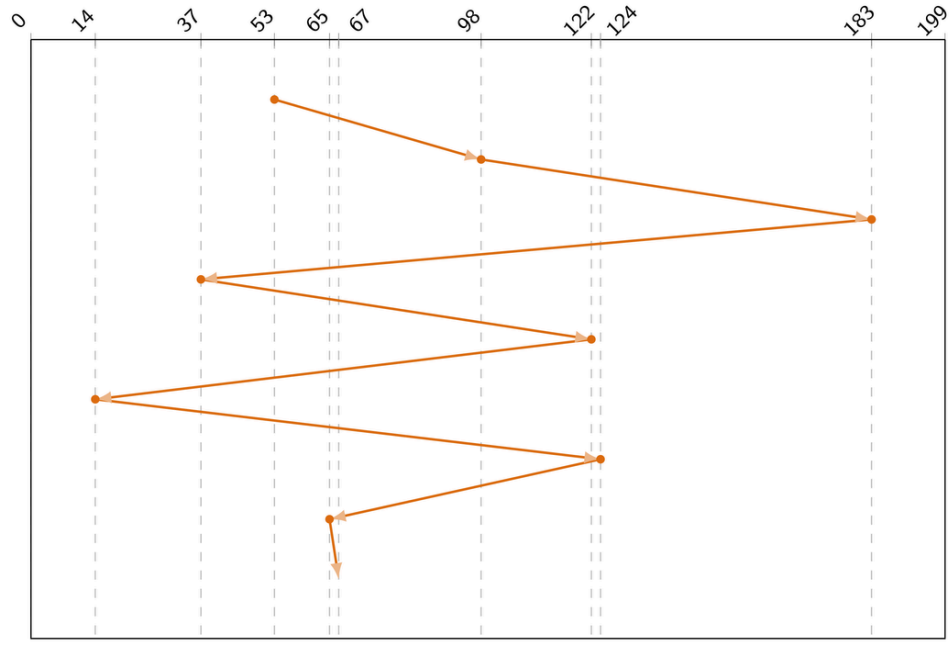
Nevýhody:

- **Nízká efektivita** – hlavičky se mohou zbytečně pohybovat sem a tam po celém disku.
- **Neoptimalizuje přístupovou dobu** – vznikají velké rozdíly mezi rychlým a pomalým požadavkem.
- **Nejhorší průměrný čas odezvy** ze všech běžně používaných metod.

Vhodnost použití:

- **Ideální pro nízkou zátěž**, kde je málo požadavků a není potřeba optimalizace výkonu.
- **Nevhodné pro vysokou zátěž** nebo servery, kde záleží na výkonu.

9. Řízení vnější paměti



Pořadí	Pozice hl.	cesta
1	53	0
2	98	45
3	189	85
4	37	146
5	122	85
6	14	108
7	124	110
8	65	59
9	67	2

Plánovací metody přístupu na disk – SSTF (Shortest Seek Time First)

SSTF vybírá ten požadavek, **který má nejkratší vzdálenost od aktuální pozice hlavičky** – tedy **nejkratší vyhledávací čas** *(seek time)*.

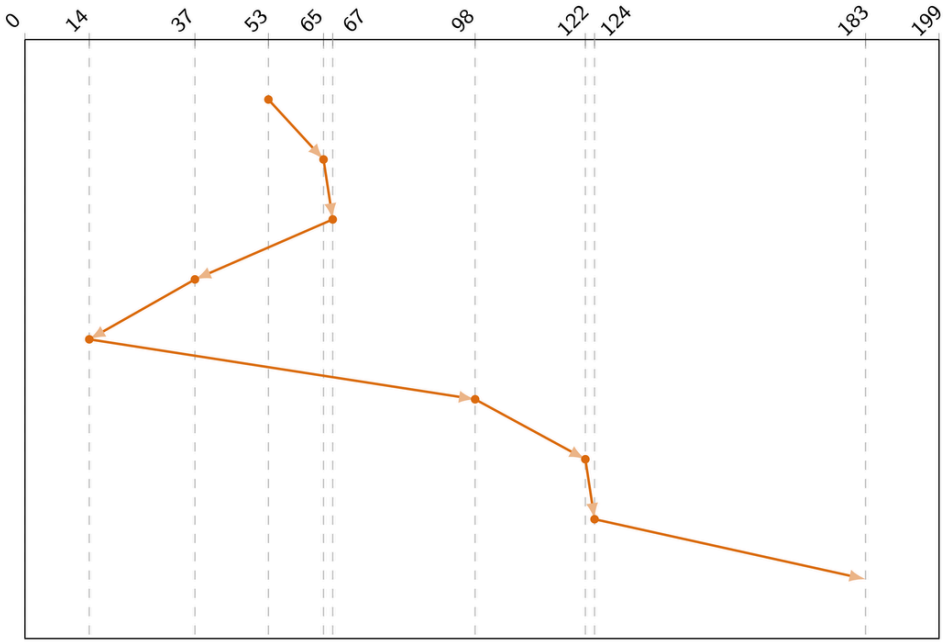
- Hlavička disku se **přesune k nejbližšímu požadovanému cylindru**, bez ohledu na pořadí přijetí požadavků.

Vlastnosti:

- **Vyšší efektivita než FCFS** – zkracuje průměrný seek time.
- **Vhodná pro krátké vzdálenosti** – minimalizuje zbytečné pohyby hlaviček.

Nevýhody:

- **Hladovění** *(starvation):*
  - Požadavky, které se nachází **daleko od aktuální pozice hlavičky, mohou být opakovaně odkládány.**
  - Hrozí, že některé požadavky **nebudou dlouho obslouženy**, pokud neustále přichází bližší požadavky.
- **Není ideální při vysoké zátěži** – zejména pokud jsou požadavky náhodně rozmístěné.



Pořadí	Pozice hl.	cesta
1	53	0
2	65	12
3	67	2
4	37	30
5	14	23
6	98	84
7	122	24
8	124	2
9	183	59

9. Řízení vnější paměti

Plánovací metody přístupu na disk – SCAN

Metoda SCAN *(přezdívaná také elevator algorithm)* simuluje **pohyb výtahu**:

- Hlavička disku se **pohybuje od jednoho kraje disku k druhému** *(např. od nejnižšího čísla cylindru k nejvyššímu)*.
- **Při cestě obsluhuje všechny požadavky, které leží na trase**.
- Po dosažení konce se směr pohybu **otočí a proces se opakuje v opačném směru**.

Chování:

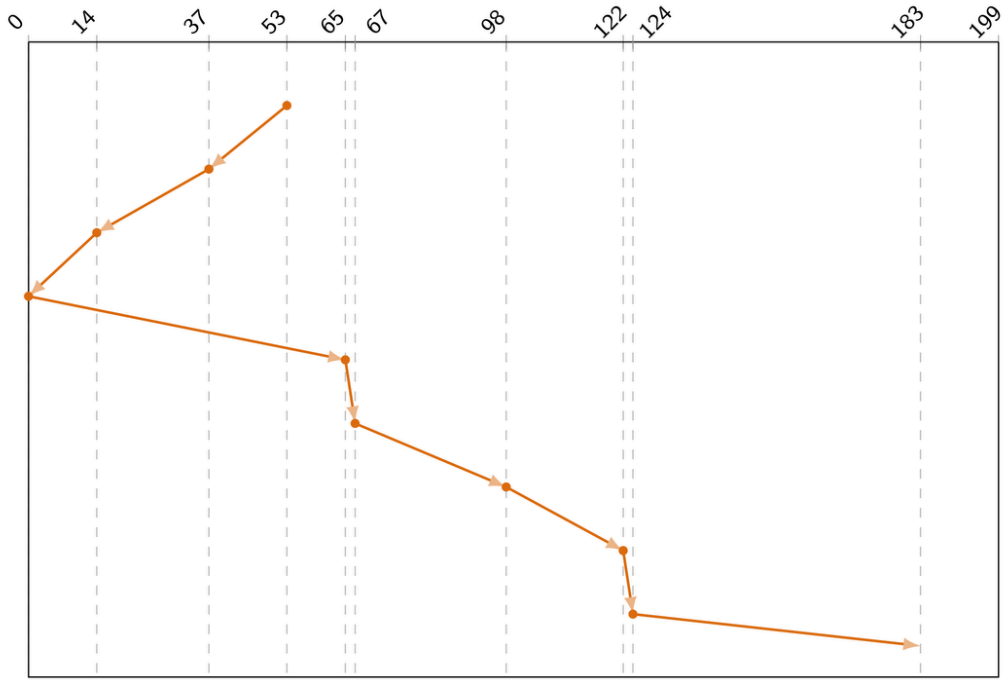
- Hlavička začíná například u **stopy 0 a jede směrem k nejvyšší stopě** *(např. 183)*.
- Obslouží všechny požadavky, které leží na její trase.
- Po dosažení konce disku se vrací zpět a obsluhuje požadavky v opačném směru.

Výhody:

- **Vyšší efektivita než FCFS a SSTF**, zejména **při větší zátěži**.
- **Žádné hladovění** – všechny požadavky budou dříve nebo později obslouženy.
- **Lepší průměrná odezva** díky organizovanému pohybu hlavičky.

Nevýhody:

- Požadavky ve středu disku jsou **obsluhovány častěji než ty na krajích**.
- V některých případech **může být čekací doba vyšší než u optimalizovanějších metod** *(např. LOOK, C-SCAN)*.



Pořadí	Pozice hl.	cesta
1	53	0
2	37	16
3	14	23
4	65	14 + 65
5	67	2
6	98	31
7	122	24
8	124	2
9	183	59

9. Řízení vnější paměti

Plánovací metody přístupu na disk – C-SCAN (Circular SCAN)

C-SCAN je **vylepšená varianta metody SCAN**, která simuluje **kruhový pohyb hlavičky disku**:

- Hlavička se pohybuje **jedním směrem** *(např. zleva doprava)* a zpracovává požadavky na trase.
- Když dorazí na **konec disku, neobsluhuje žádné požadavky cestou zpět**.
- Místo toho se **rychle vrátí na začátek disku** *(např. stopa 0)* **bez obsluhy**, a tam **pokračuje ve zpracování** dalších požadavků ve stejném směru.

Chování:

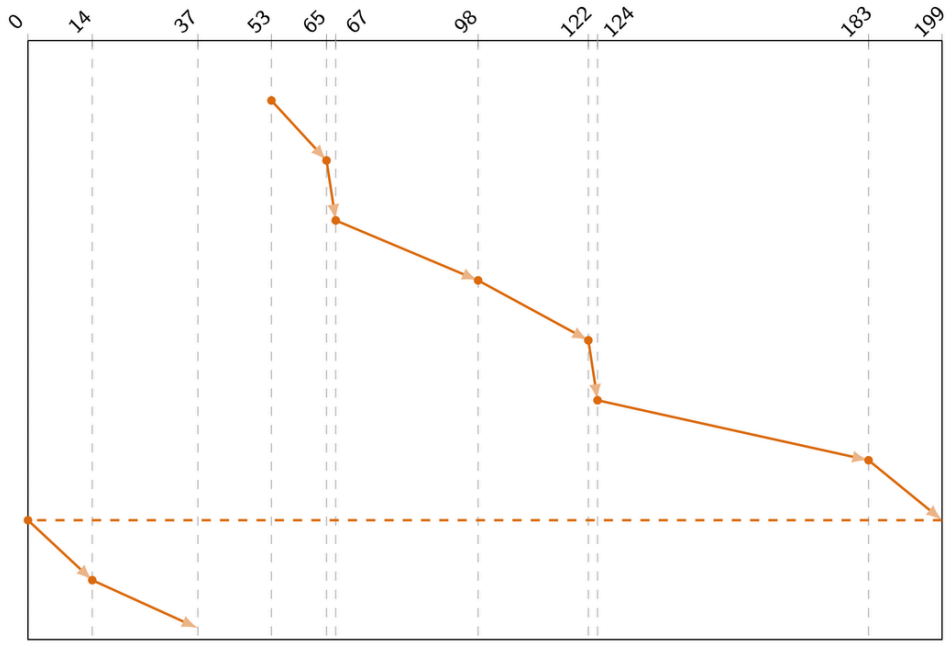
- **Např. zpracovává požadavky od stopy 0 ke stopě 199.**
- Po **dosažení konce** *(199)* se rychle **vrátí na začátek** *(0)* – bez zpracování požadavků cestou zpět.
- **Proces se opakuje** – pouze jednosměrné obsluhování požadavků.

Výhody:

- **Rovnoměrný přístup ke všem požadavkům** – žádná část disku není upřednostňována *(na rozdíl od SCAN, kde jsou střední stopy obsluhovány častěji).*
- **Žádné hladovění** – každý požadavek bude dříve nebo později obsloužen.
- **Stabilní odezva** – vhodné pro systémy, kde je důležitá rovnováha přístupu.

Nevýhody:

- **Zpětný pohyb hlavičky** *(z konce na začátek)* **je neproduktivní** – žádná obsluha požadavků při návratu.
- O něco **vyšší průměrný seek time než u SCAN** v některých scénářích.



Pořadí	Pozice hl.	cesta
1	53	0
2	65	12
3	67	2
4	98	31
5	122	24
6	124	2
7	183	59
8	14	30 (229)
9	37	23

9. Řízení vnější paměti

Plánovací metody přístupu na disk – LOOK

LOOK je **vylepšená verze algoritmu SCAN**, která se chová chytřeji při pohybu hlavičky:

- Hlavička neprojíždí celý disk od kraje ke kraji jako u SCAN.
- Místo toho se **pohybuje pouze k nejvzdálenějšímu aktivnímu požadavku v daném směru a poté změní směr.**

Chování:

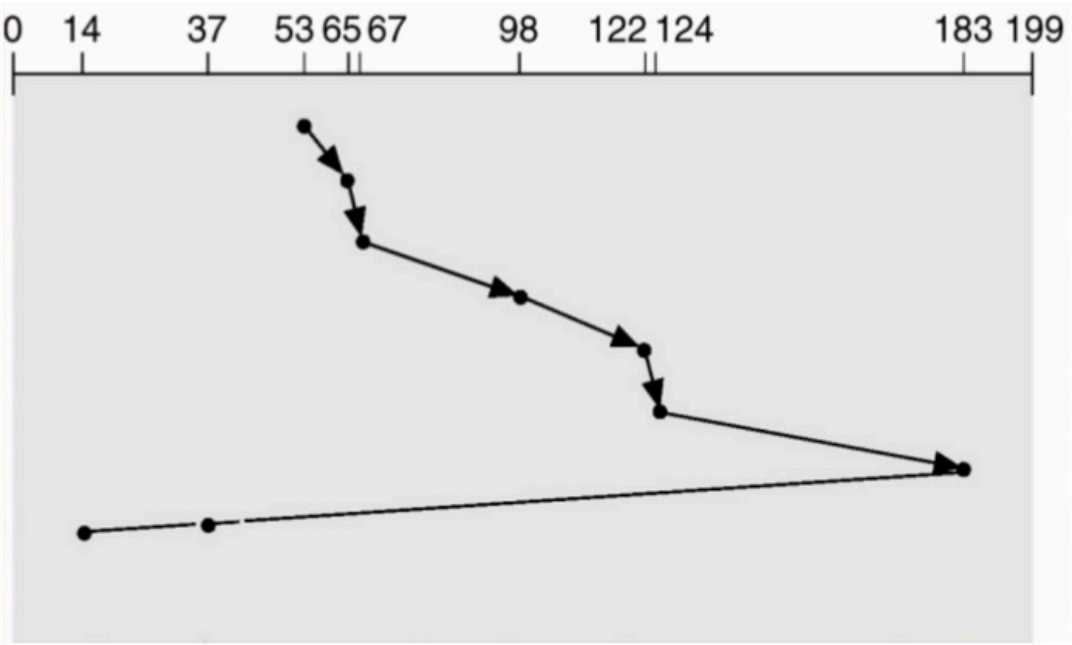
- Hlavička disku se vydá směrem k **nejbližšímu požadavku v daném směru**.
- **Obsluhuje všechny požadavky po cestě**.
- Jakmile dorazí **k poslednímu požadavku v tomto směru, otočí se a pokračuje zpět, opět obsluhuje požadavky po cestě**.
- Na rozdíl od SCAN **nezajíždí zbytečně až na kraj disku**, pokud tam nečeká žádný požadavek.

Výhody:

- **Efektivnější než SCAN** – eliminuje zbytečné pohyby hlavičky na konec disku.
- **Rychlejší reakční doba**, pokud jsou požadavky soustředěny blízko sebe.
- **Žádné hladovění** – všechny požadavky budou obslouženy.

Nevýhody:

- Stejně jako SCAN může **častěji obsluhovat střední oblasti disku než okrajové**.
- **Složitější na implementaci** než FCFS nebo SSTF.



Pořadí	Pozice hl.	cesta
1	53	0
2	65	12
3	67	2
4	98	31
5	122	24
6	124	2
7	183	59
8	37	146
9	14	23

9. Řízení vnější paměti

Plánovací metody přístupu na disk – C-LOOK (Circular LOOK)

C-LOOK je **kruhová varianta algoritmu LOOK**, která funguje **podobně jako C-SCAN,** ale je ještě efektivnější:

- **Hlavička se pohybuje pouze mezi aktivními požadavky v jednom směru** – od **nejnižšího k nejvyššímu.**
- Po dosažení **posledního požadavku** v **tomto směru se vrátí na začátek** *(k nejnižšímu požadavku)*, bez obsluhy požadavků **cestou zpět.**
- Proces se pak **opakuje**.

Chování:

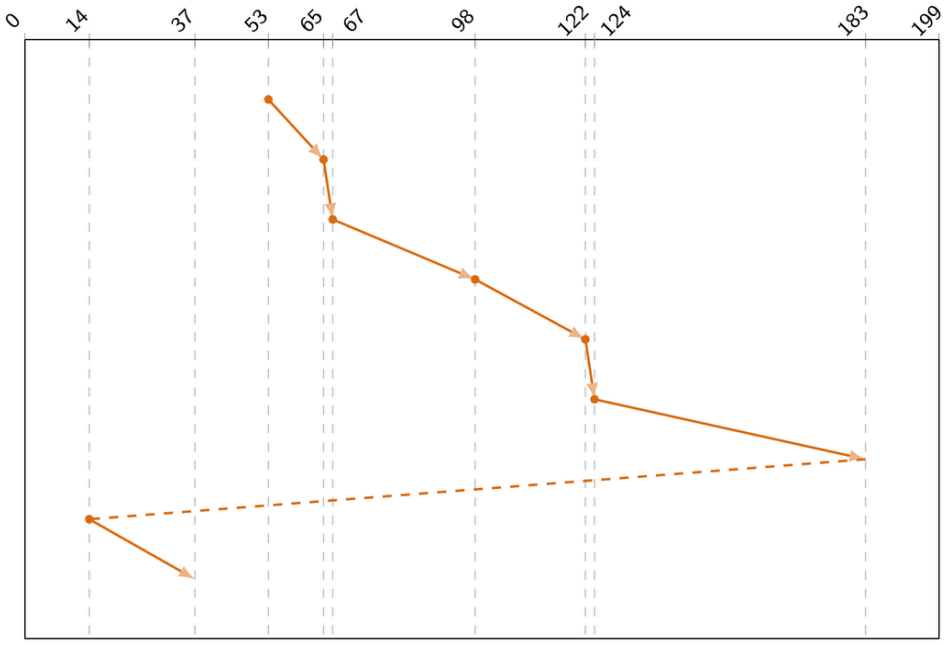
- Hlavička začíná u **nejnižšího požadavku a jede směrem ke nejvyššímu**, obsluhuje všechny požadavky po cestě.
- Po dosažení **posledního požadavku se rychle přesune zpět na začátek** *(první požadavek)*.
- Tento cirkulární pohyb se **neustále opakuje** – vždy jedním směrem.

Výhody:

- **Rovnoměrný přístup ke všem požadavkům** – žádná část disku není upřednostňována.
- **Efektivnější než SCAN a LOOK**, protože se neplýtvá časem obsluhou cestou zpět.
- **Rychlejší odezva** pro požadavky u začátku disku po návratu hlavičky.

Nevýhody:

- **Nepřirozený pohyb** *(skok zpět na začátek)* může být **nevhodný v systémech s požadavkem na reálný čas**.
- Návrat hlavičky **je neproduktivní** *(jako u C-SCAN)*, ale **kratší**, protože nejde až na kraj disku.



Pořadí	Pozice hl.	cesta
1	53	0
2	65	12
3	67	2
4	98	31
5	122	24
6	124	2
7	183	59
8	14	169
9	37	23

9. Řízení vnější paměti

Hard Disk Drive (HDD) vs. Solid State Drive (SSD)

HDD – Hard Disk Drive

Pevný disk je **mechanické zařízení**, které **ukládá data na magneticky pokryté rotující plotny**.

Vlastnosti:

- Magnetická vrstva na rotujících plotnách.
- Mechanické části – vystavovací mechanismus, hlavy, plotny.
- Data jsou čtena/zapisována mechanicky pomocí pohybu hlav.

Nevýhody:

- **Vyšší přístupový čas** *(typicky 5–15 ms)* → pomalejší než SSD.
- **Vyšší spotřeba elektrické energie** kvůli pohyblivým částem.
- **Fragmentace dat zhoršuje výkon** – čtení z různých míst je pomalejší.
- **Náchylnost k poškození** – vibrace, nárazy mohou poškodit plotny nebo hlavičky.
- **Hlučný provoz** – slyšitelné šumění a klikání.

SSD – Solid State Drive

SSD je **elektronické zařízení bez pohyblivých částí**, které ukládá data do paměťových čipů *(flash paměť)*.

Výhody:

- **Nízký přístupový čas** *(typicky pod 1 ms)* → mnohem rychlejší než HDD.
- **Žádné pohyblivé části** → odolnější vůči otřesům a nárazům.
- **Nezávislé ukládání dat** – nedochází k fragmentaci.
- **Tichý provoz** – žádné mechanické zvuky.
- **Nižší spotřeba** energie.

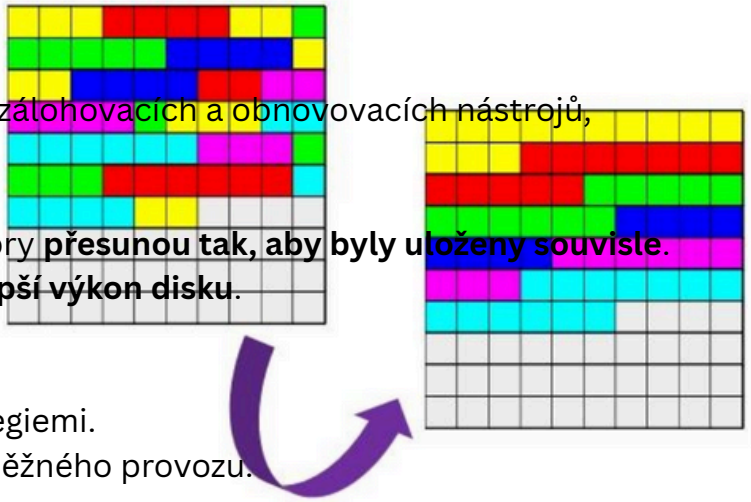
Nevýhody:

- **Vyšší cena** za 1 GB než u HDD.
- **Omezený počet zápisových cyklů** – ale dnes díky technologiím *(TRIM, wear leveling)* už většinou nevadí.

Fragmentace a Defragmentace

Fragmentace

- Fragmentovaný soubor je **uložen roztříštěně po disku** – není uložen v sousedních *(souvislých)* clusterech.
- Data jsou uložena na **různých cylindrech, což způsobuje pomalé načítání** *(hlavička se musí přesouvat)*.
- **Vzniká při častém mazání a vytváření nových souborů**, které se nevejdou do uvolněných mezer.



Důsledky:

- **Pomalejší čtení souborů**, Snížená účinnost zálohovacích a obnovovacích nástrojů, Celkově nižší výkon disku.

Defragmentace

- Proces, při kterém se fragmentované soubory **přesunou tak, aby byly uloženy souvisle**.
- Výsledkem je **rychlejší přístup k datům a lepší výkon disku**.

Nástroje pro defragmentaci:

- Integrovaná defragmentace v OS Windows.
- **O&O Defrag** – pokročilý nástroj s více strategiemi.
- **Diskeeper** – optimalizace výkonu i během běžného provozu.