



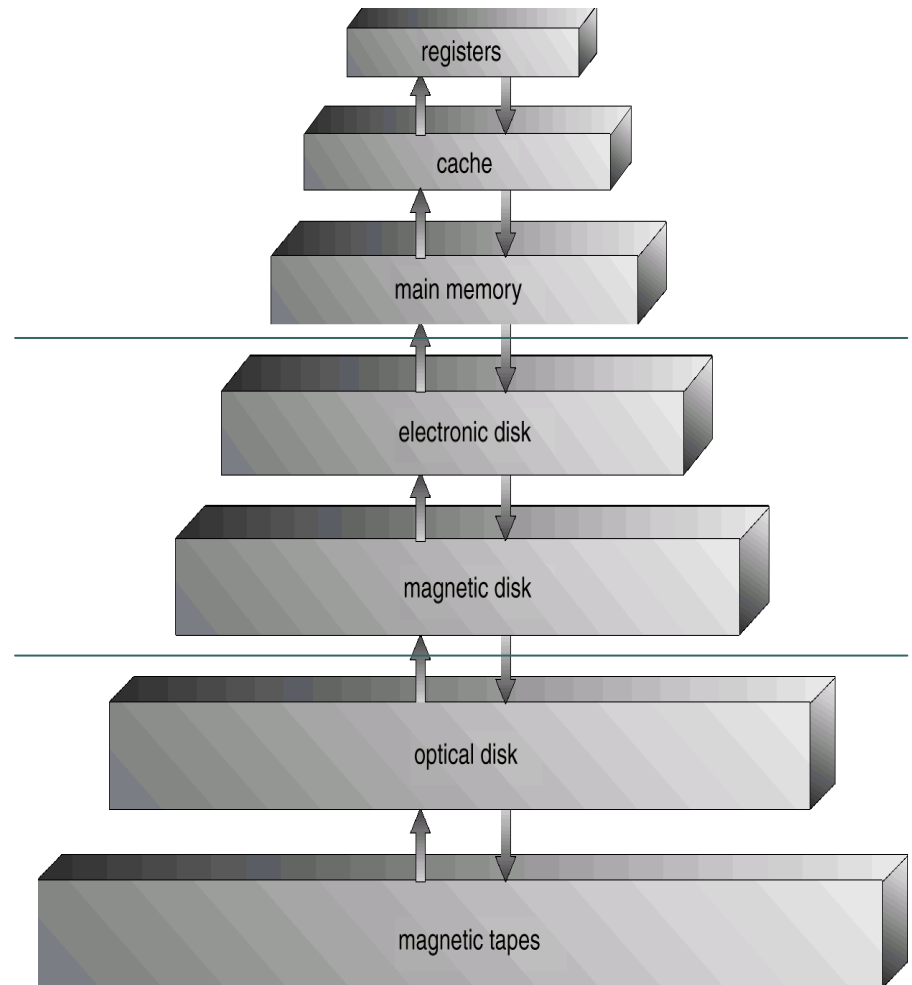
PB153

Operační systémy a jejich rozhraní

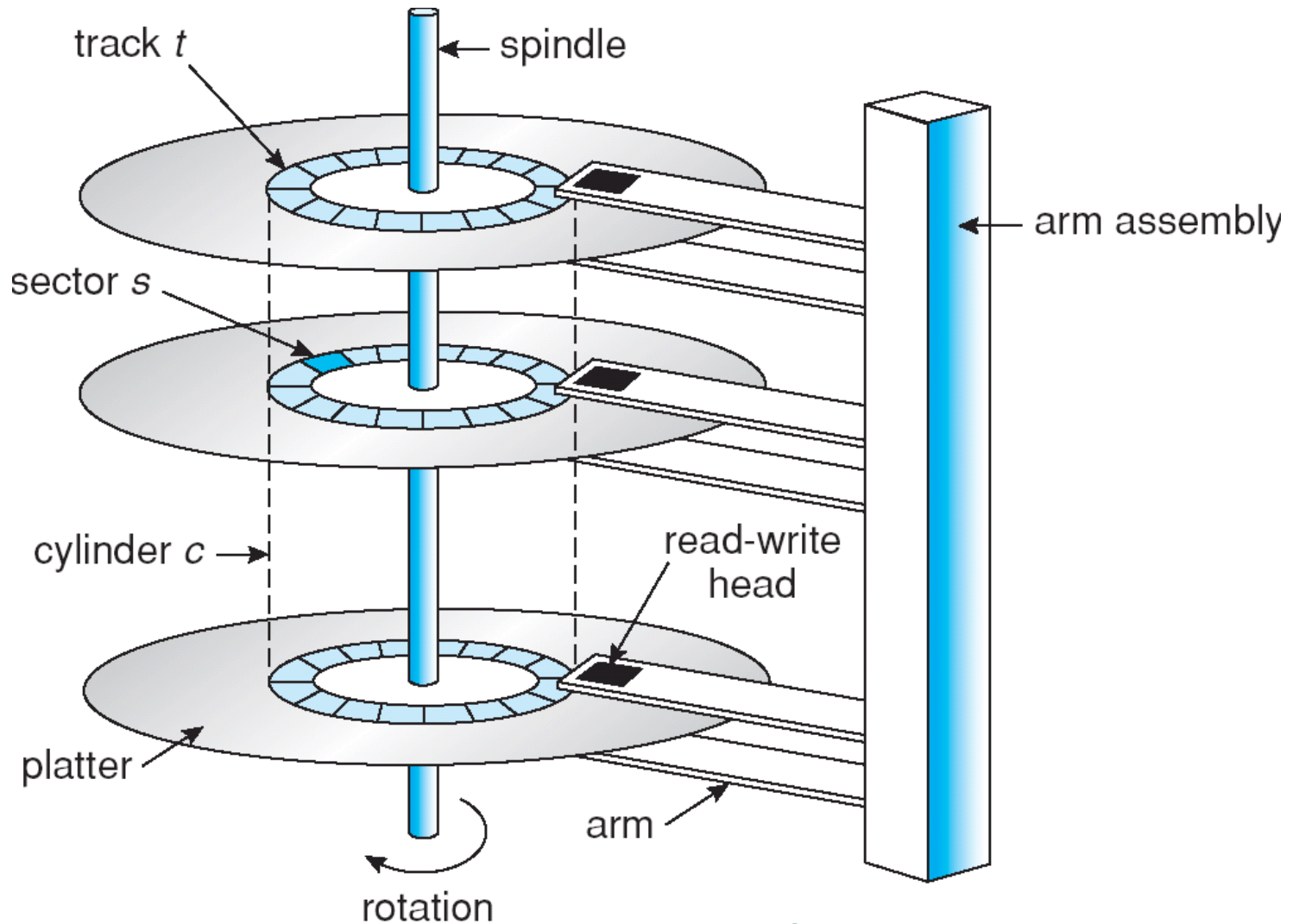
Vnější paměti

Paměťová hierarchie

- Primární paměti
 - nejrychlejší
 - energeticky závislé
 - Cache, hlavní (operační) paměť
- Sekundární paměti
 - středně rychlé
 - energeticky nezávislé
 - Také nazývané „on-line storage“
 - flash disky, magnetické disky
- Terciální paměti
 - levná typicky vyměnitelná média
 - pomalé
 - energeticky nezávislé
 - také nazývané „off-line storage“
 - floppy disky, magnetické pásky, optické disky



Magnetické disky





Struktura disku

- Diskové mechanismy se adresují jako velká 1-dimensionální pole logických bloků
 - logické bloky jsou nejmenší jednotkou přenosu dat
- 1-dimensionální pole logických bloků je zobrazováno do sektorů disku sekvenčně
 - sektor 0
 - první sektor na první stopě vnějšího cylindru
 - zobrazování pokračuje po této stopě, potom po ostatních stopách tohoto cylindru, a potom po cylindrech směrem ke středu



Plánování disku

- OS je odpovědný za efektivní používání hardware
 - pro disky: co nejrychlejší přístup a co největší šířka pásma
- Doba přístupu (*access time*) je dána:
 - dobou vystavení (*seek time*) – na cylindr se stopou s adresovaným sektorem
 - dobou rotačního zpoždění – dodatečná doba do průchodu adresovaného sektoru pod čtecí/zápisovou hlavou
- Minimalizace doby vystavení
 - doba vystavení \approx vystavovací vzdálenosti
 - řeší plánování činnosti disku
- Rotační zpoždění
 - shora omezeno konstantou
- Šířka pásma
 - počet přenesených bytů / doba od zadání skupiny požadavků do jejich ukončení
 - převzatý pojem z telekomunikací

Plánování disku

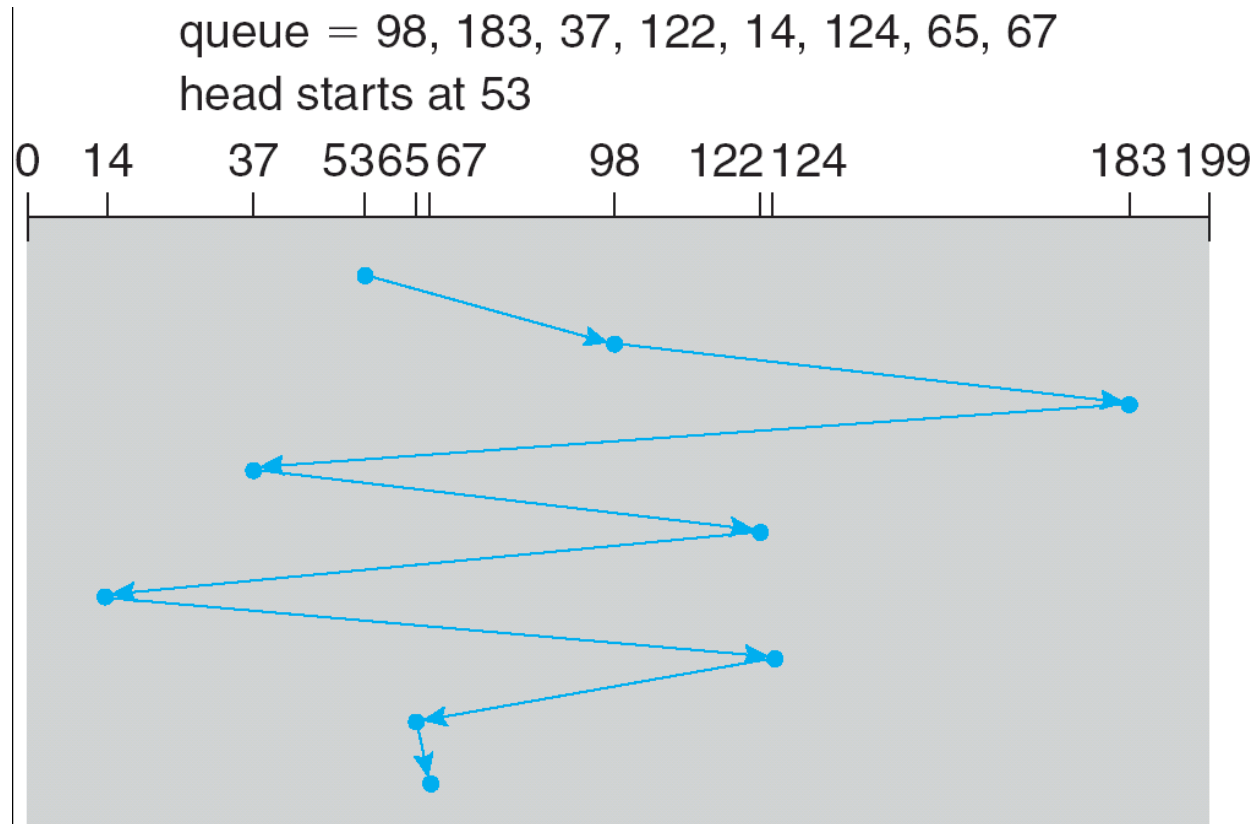
- Existuje celá řada algoritmů pro plánování přístupu na disk.
 - „Disk scheduling“
- Příklad: vzorová fronta požadavků na přístup k disku (máme cylindry 0-199).

98, 183, 37, 122, 14, 124, 65, 67

Hlavička disku vystavena na pozici 53

FCFS

- Celkem přesun o 640 cylindrů





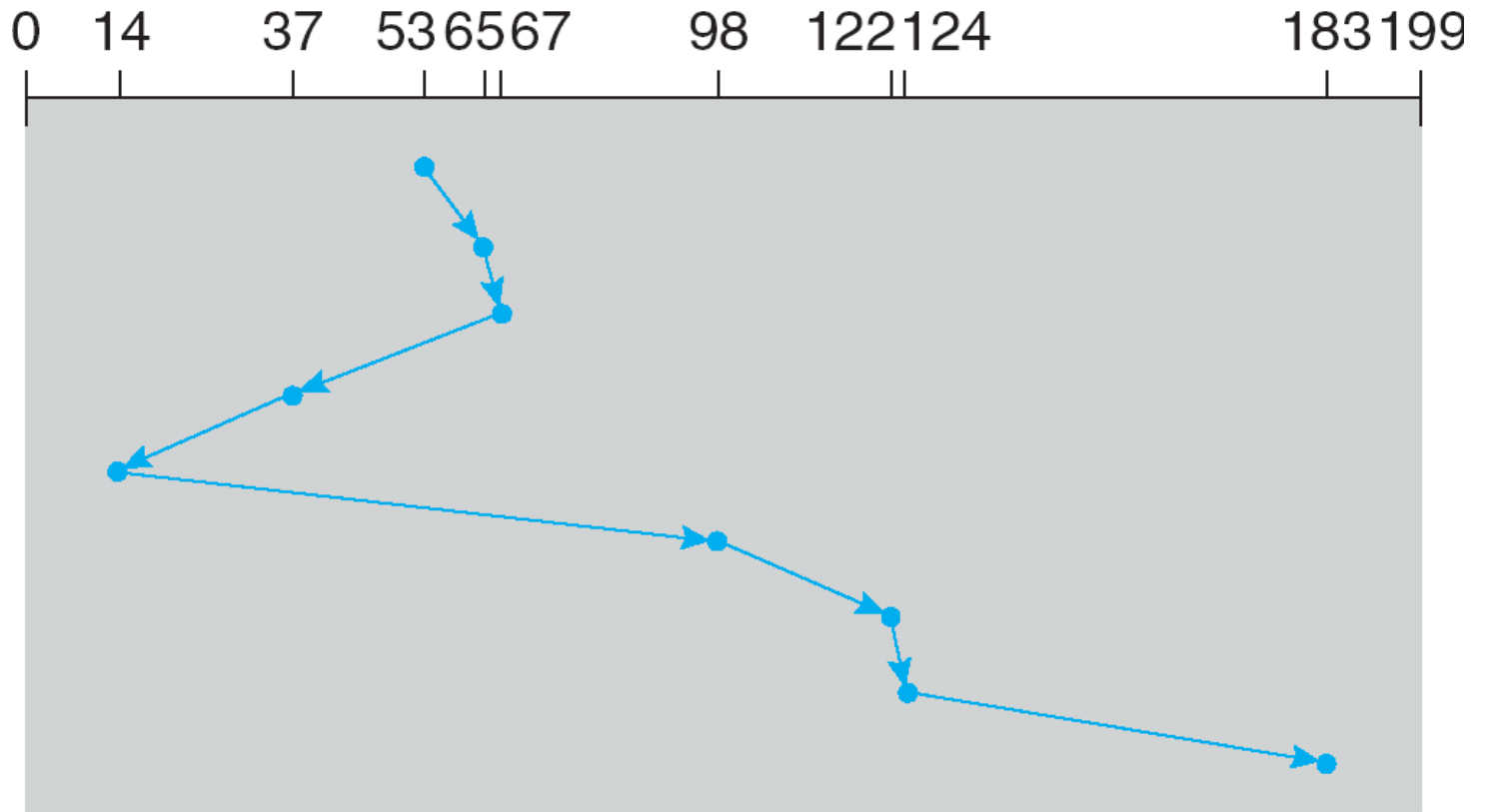
SSTF

- Z fronty požadavků vybírá ten požadavek, který vyžaduje minimální dobu vystavení od současné pozice hlavičky
- SSTF (*shortest seek time first*) algoritmus je variantou algoritmu SJF (*shortest job first*); může způsobit stárnutí požadavků.
- Náš příklad vyžaduje přesun o 236 cylindrů

SSTF

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53





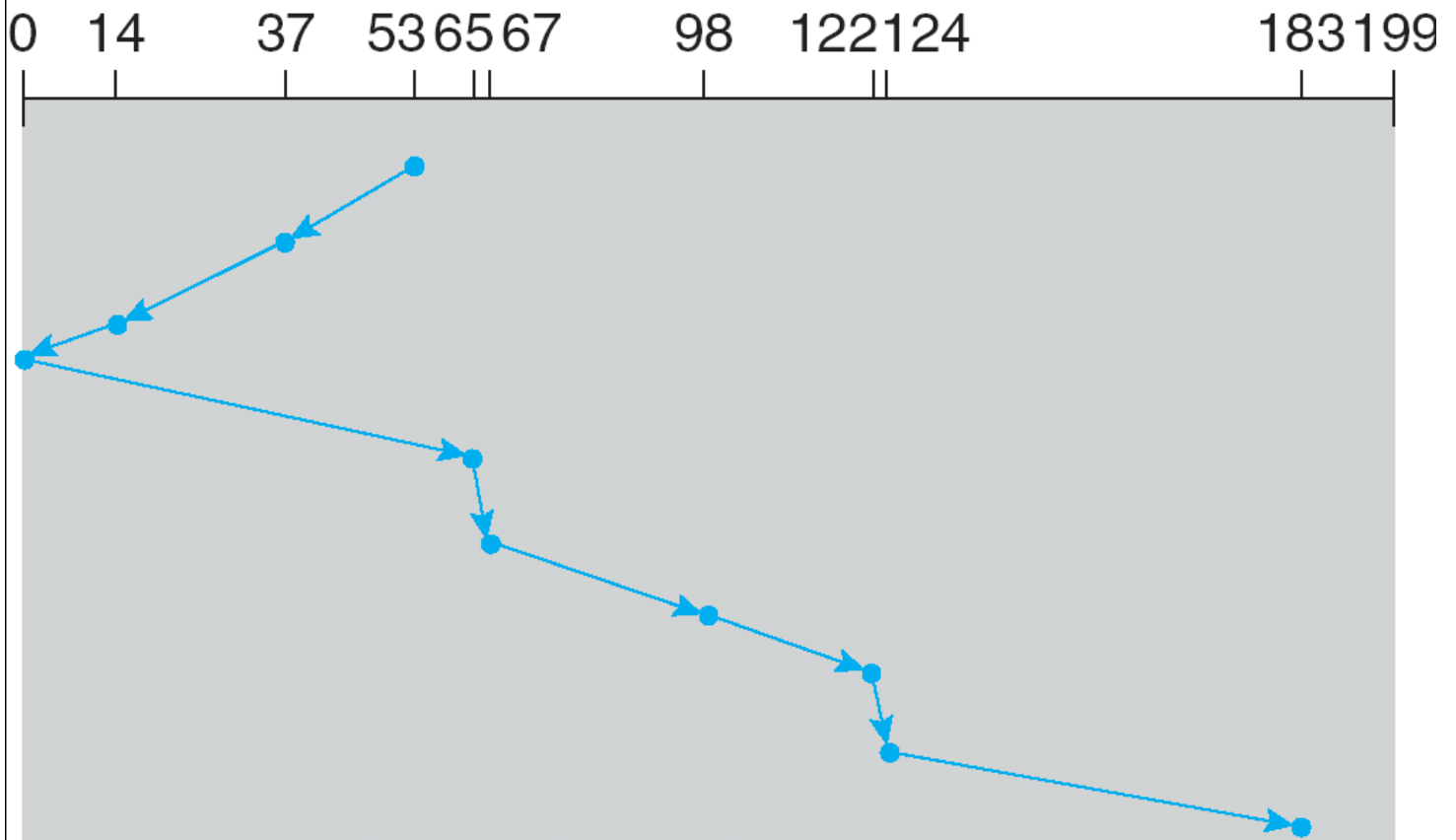
SCAN

- Hlavička disku začíná na jedné straně disku a přesunuje se při splňování požadavků ke druhé straně disku. Pak se vrací zpět a opět plní požadavky.
- Někdy nazývané *algoritmus typu výtah (elevator)*.
- Náš příklad vyžaduje přesun o 208 cylindrů.

SCAN

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53





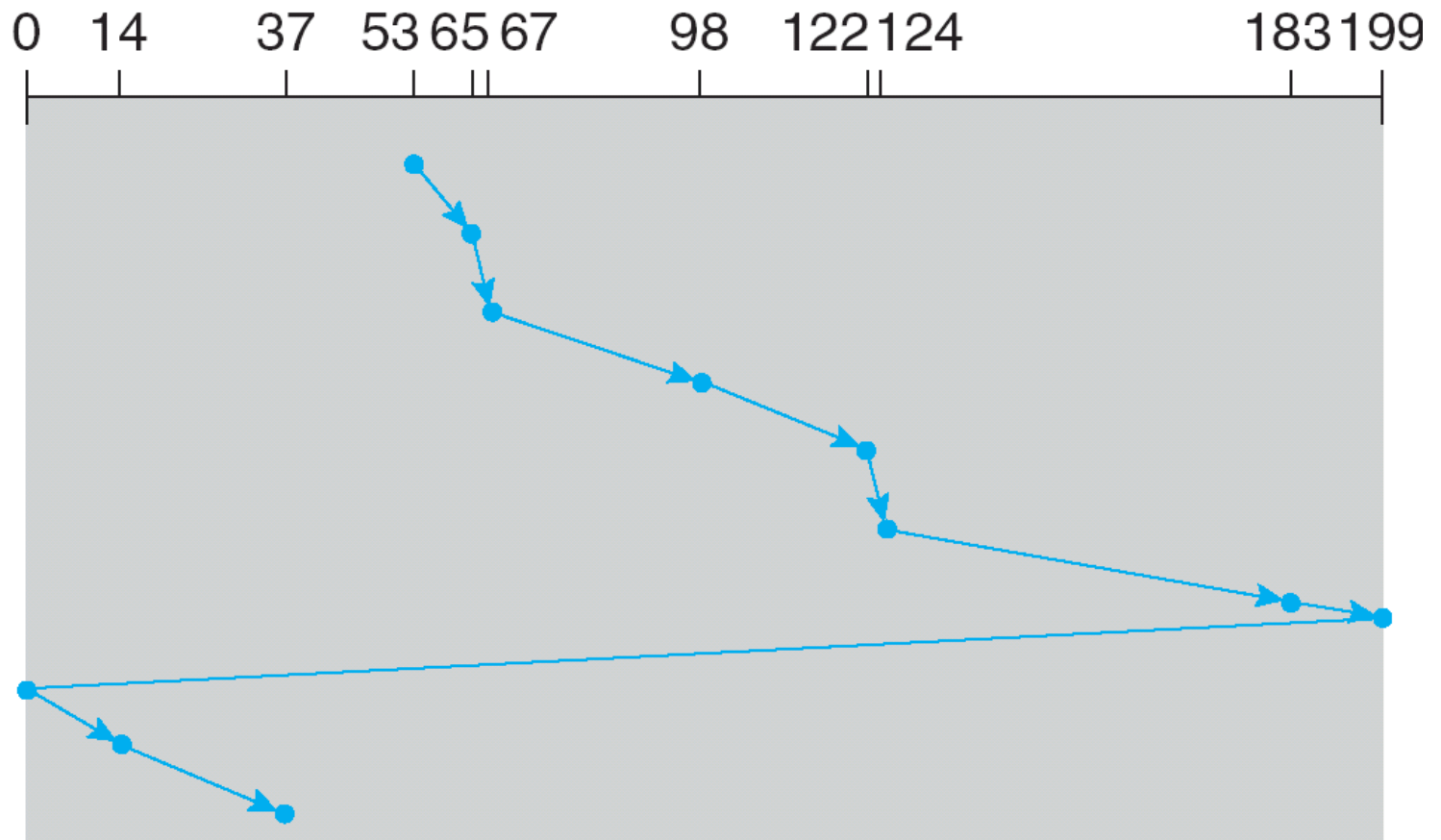
C-SCAN

- Poskytuje jednotnější čekací dobu než SCAN
- Hlavička se posouvá z jednoho konce disku na druhý a zpracovává požadavky. Potom se vrací zpět bez vyřizování požadavků a opět začíná vyřizovat požadavky z prvního konce.
- Cylindry považuje za kruhový seznam, který za posledním cylindrem pokračuje opět prvním cylindrem.

C-SCAN

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53





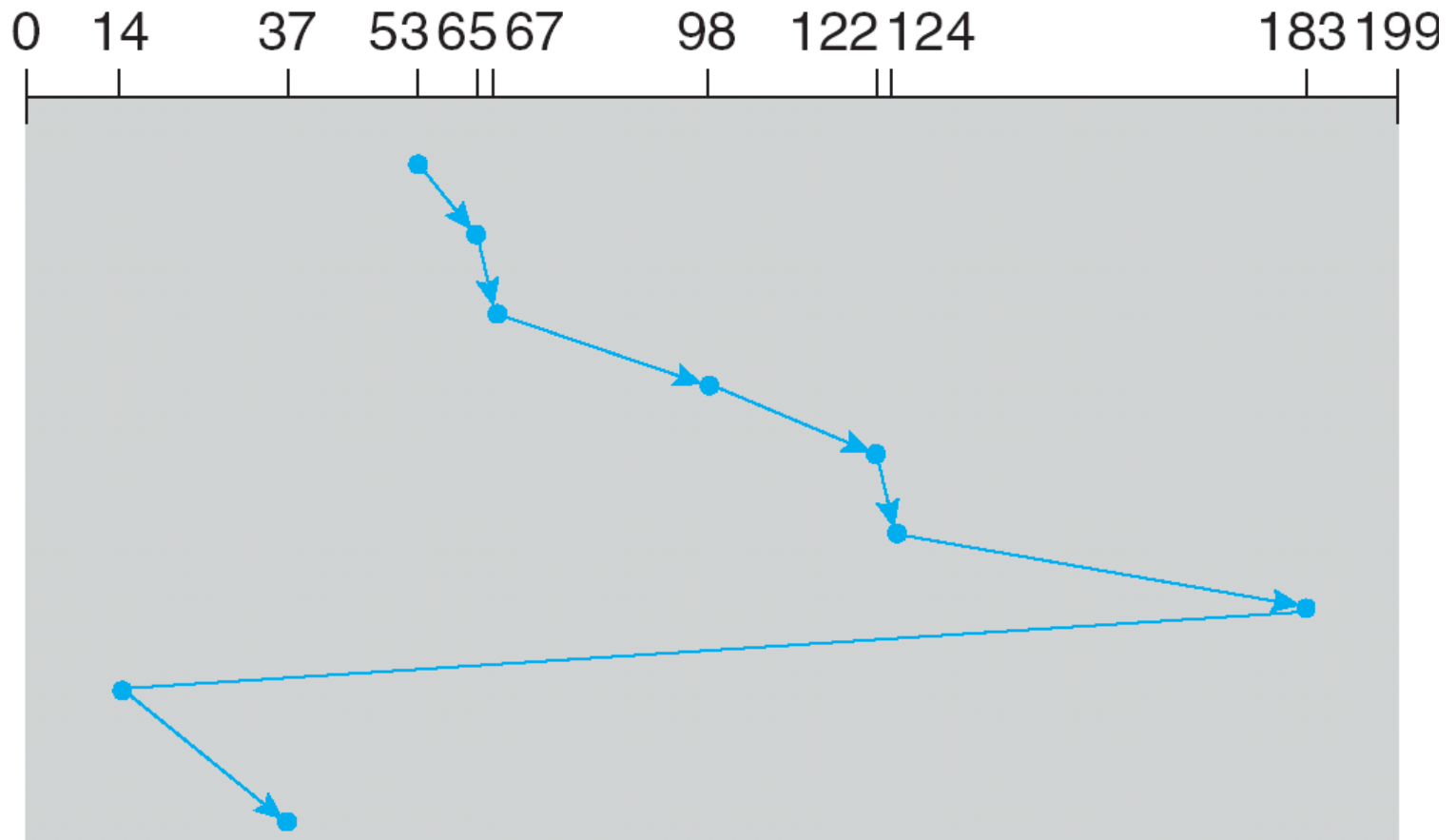
C-LOOK

- Obdoba C-SCAN, ale hlavička jen potud do kraje, pokud existují požadavky.
- Pak se vrací zpět

C-LOOK

queue 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



Výběr algoritmu

- SSTF je přirozený, má přirozené chápání
- SCAN a C-SCAN jsou vhodnější pro těžkou zátěž disku
- Výkon závisí na počtu a typech požadavků
- Požadavky na disk mohou být ovlivněny metodami organizace souborů v souborovém systému
- Plánovací algoritmus by měl být napsán jako samostatný modul, aby plánovací algoritmus OS bylo možné zaměňovat
- Častá implicitní volba bývá SSTF nebo LOOK

Moderní HW

- U moderních disků nemusí být známé mapování logických bloků na fyzické adresy
- Disku předáme skupinu požadavků a disk si pořadí optimalizuje sám
- OS přesto může mít zájem na vlastním řazení požadavků
 - priorita I/O operací z důvodu výpadků stránek
 - Pořadí operací zápisu dat a metadat souborového systému



Příklad: Linux

- K dispozici několik plánovacích algoritmů

```
# cat /sys/block/sda/queue/scheduler  
noop anticipatory deadline [cfq]
```

```
# echo noop >/sys/block/sda/queue/scheduler
```

- Fronty v řadiči disku mohou měnit pořadí od OS
 - Fronta na desítky až stovky požadavků
- Algoritmy
 - Noop
 - Anticipatory
 - Deadline
 - CFQ (Completely Fair Queue)



NOOP

- Nemění pořadí požadavků
- Jen pokud nově příchozí navazuje na předchozí požadavek, požadavky budou sloučeny
 - To stejně obvykle řeší už vrstva blokového zařízení nebo vrstva souborového systému.



Deadline

- Vychází z algoritmu SCAN
- Přidává deadline (do kdy má být požadavek vyřízen)
- Po dávce požadavků se vzrůstajícími čísly sektorů (SCAN) kontroluje deadline
 - Pokud je třeba vytvoří speciální dávku pro vyřešení požadavků s expirovaným deadline.
- Upřednostňuje čtení před zápisem

Anticipatory scheduler (AS)

- Jako deadline, ale přidává očekávání, že při sekvenčním přístupu k souboru přijde po jednom požadavku brzy požadavek následující. Proto chvíli vyčká (opravdu) ...
- Umožňuje i krátké přesuny zpět
 - Penalizuje dvojnásobnou vzdáleností
- Neodlišuje požadavky čtení a zápisu
- Odlišuje asynchronní požadavky od synchronních

Completely Fair Queuing

- Snaží se být spravedlivý k *procesům*
- Každý proces dostává určitý časový díl (slice) kdy má exkluzivní přístup pro synchronní požadavky
- Parametry
 - slice_sync – délka slice v ms (bere v úvahu I/O prioritu procesu)
 - quantum – počet požadavků
- 17 front (pro každou prioritu jedna) pro asynchronní požadavky
- Každá fronta získává určitý slice a fronty jsou obsluhovány algoritmem RR



CFQ parametry

Tunable	Default value	Purpose
<code>back_seek_max</code>	16384	(KiB) Max. backward seek
<code>back_seek_penalty</code>	2	reverse seek bias
<code>fifo_expire_sync</code>	125	(ms) deadline for synchronous requests
<code>fifo_expire_async</code>	250	(ms) deadline for async requests
<code>quantum</code>	4	Maximum requests to service in each batch in each round
<code>slice_async</code>	40	(ms) base time slice for async. requests
<code>slice_sync</code>	100	(ms) base time slice for sync. requests
<code>slice_async_rq</code>	2	base number of async requests per round
<code>slice_idle</code>	8	(ms) maximum idle time between requests

I/O priorita procesů

- Příkaz `ionice` (od 2.6.13 pro CFQ)

NAME

`ionice` - get/set program io scheduling class and priority

SYNOPSIS

```
ionice [-c] [-n] [-p] [COMMAND [ARG...]]
```

Scheduling class	Number	Possible priority
real time	1	8 priority levels are defined denoting how big a time slice a given process will receive on each scheduling window
best-effort	2	0-7, with lower number being higher priority
idle	3	Nil (does not take a priority argument)

Rychlost terciálních pamětí

- Dva aspekty
 - šířka pásma, bandwidth
 - zpoždění, latency
- Šířka pásma se měří v B/s
 - podporovaná šířka pásma
 - průměrná rychlost přenosu dat během velkého přenosu
 - počet B / doba přenosu
 - efektivní šířka pásma
 - Průměr za celou dobu I/O operace, vč. vystavení, nalezení umístění, přepnutí svazku atd.

Rychlost terciálních pamětí

- Zpoždění při přístupu
 - doba potřebná pro vyhledání dat
 - na disku – vystavení, natočení, určitě < 35 ms
 - na pásce – vč. přetočení pásky na požadovaný blok, desítky až stovky sekund
 - pásy jsou typicky 1000x pomalejší než disky
- Nízká cena terciární paměti je dána tím, že se pracuje s mnoha levnými svazky v malém počtu drahých mechanismů
- Vyměnitelná knihovna se nejlépe využije pro ukládání řídké používaných dat
 - uspokojuje se malý počet I/O požadavků



Spolehlivost

- Pevný disk je *asi* spolehlivější než vyměnitelný disk nebo páska
- Optický disk je *asi* spolehlivější než magnetický páska
- Padnutí hlav v pevném disku *obvykle* znamená zničení dat
- Porucha pásky nebo optického disku *obvykle* neznámá zničení všech dat



Cena

- Operační paměť je mnohem dražší než disková paměť
- Cena MB na pevném disku je srovnatelná s cenou MB na pásce (dříve pásy levnější)
- Kapacity magnetických pásek nedržely v posledních letech krok s nárůstem kapacit pevných disků
- Terciální paměť může ušetřit peníze pouze když se používá mnohem více svazků než mechanismů



Technologie RAID

- RAID: Redundant Arrays of Independent (Inexpensive) Disks
 - organizace disků řízená tak, že poskytuje objem jednoho disku
 - s velkou kapacitou a rychlostí díky tomu, že mnoho disků pracuje paralelně
 - s velkou spolehlivostí, data se uchovávají redundantně, lze je obnovit i po poruše některého z disků
- Pravděpodobnost, že některý disk z množiny N disků selže je mnohem vyšší, než pravděpodobnost, že selže jediný disk
 - $N = 100$ disků, každý má MTTF = 100 000 hodin (cca 11 let), celý systém bude mít MTTF = 1000 hodin (cca 41 dní)
 - techniky na bázi redundance chránící před ztrátou dat jsou pro systémy s velkým počtem komponent (disků) kritické
- Původní záměr
 - levná alternativa nahrazující velké drahé disky
 - „I“ je interpretováno jako „independent“

RAID: zvýšení spolehlivosti

- Redundance
 - nadbytečnost, doplňková informace použitelná pro obnovu informace po poruše (disku)
- Zrcadlení (stínování), Mirroring (shadowing)
 - každý disk je duplikován, 1 logický disk je tvořen 2 fyzickými disky
 - každý zápis se provede na obou discích, čte se z jednoho disku
 - jestliže se jeden disk porouchá, data jsou k dispozici na druhém disku
 - ke ztrátě dat dojde při výpadku obou disků, když zrcadlový disk selže dříve, než se systém opraví
 - průměrná doba do ztráty dat závisí na průměrné době do poruchy a průměrné doby opravy
 - Např. MTTF = 100 000 hodin, průměrná doba opravy 10 hodin, dává u zrcadlené dvojice disků průměrnou dobu ztráty dat $500 \cdot 10^6$ hodin (čili 57 000 let), když budeme ignorovat požáry apod.

RAID: zvýšení výkonu

- Dva hlavní cíle paralelismu v diskových systémech
 - zvýšení propustnosti vyvážením zátěže malými přístupy
 - paralelizace velkých přístupů s cílem zkrácení doby odpovědi
- Zvýšení přenosové rychlosti paralelním zápisem do více disků (dělení, striping)
 - bit-level striping
 - dělení bitů každého bytu mezi samostatné disky
 - v poli 8 disků se zapisuje bit i každého bytu na disk i
 - čtení dat probíhá 8x rychleji než z jednoho disku
 - vystavení je delší než v případě jednoho disku
 - dnes se bit-level striping de facto už nepoužívá
 - blok-level striping
 - systém s n disky, blok souboru i se zapisuje na disk $(i \bmod n) + 1$
 - požadavky na různé bloky se mohou realizovat paralelně, jestliže bloky leží na různých discích
 - požadavek na dlouhou posloupnost bloků může použít všechny disky paralelně

Úrovně RAID, přehled

RAID Level 0: Žádná redundance, jen souběžnost

RAID Level 1: Spolehlivost dosažená zrcadlením disků

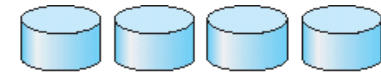
RAID Level 2: Hamming code error correction

RAID Level 3: 1 kontrolní disk na skupinu, dělení bitů

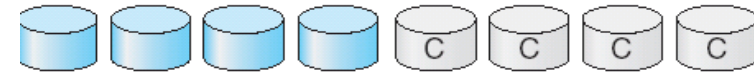
RAID Level 4: Nezávislé operace read/write, dělení bloků

RAID Level 5: Data/parity přes všechny disky (více souběžný přístup)

RAID Level 6: Odolnost při více než jedné poruše disku



(a) RAID 0: non-redundant striping.



(b) RAID 1: mirrored disks.



(c) RAID 2: memory-style error-correcting codes.



(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.

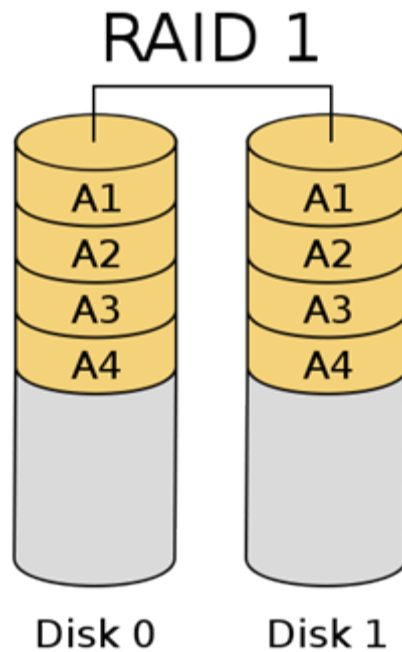
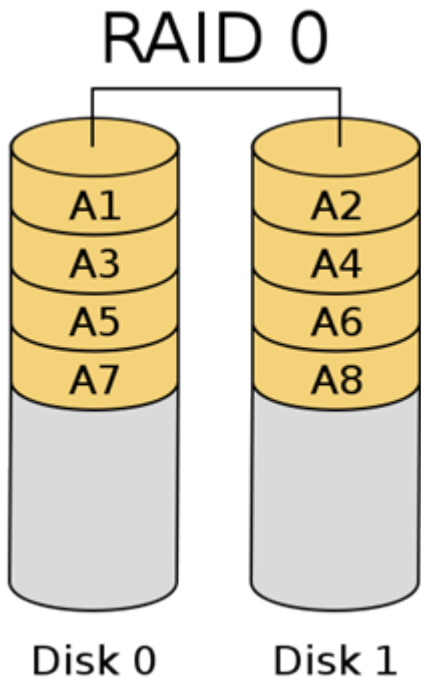


(f) RAID 5: block-interleaved distributed parity.



(g) RAID 6: P + Q redundancy.

Běžně používaný RAID

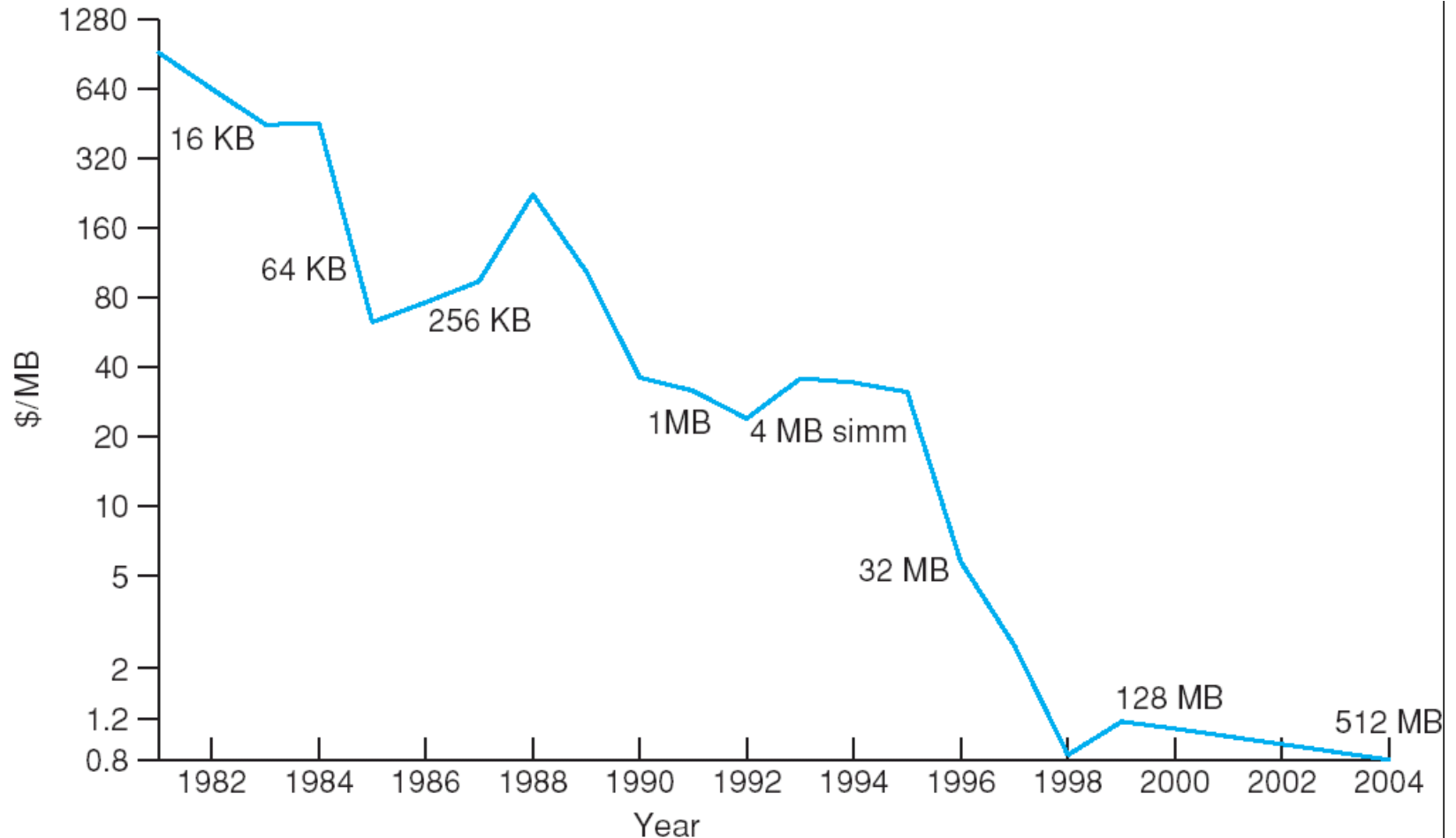




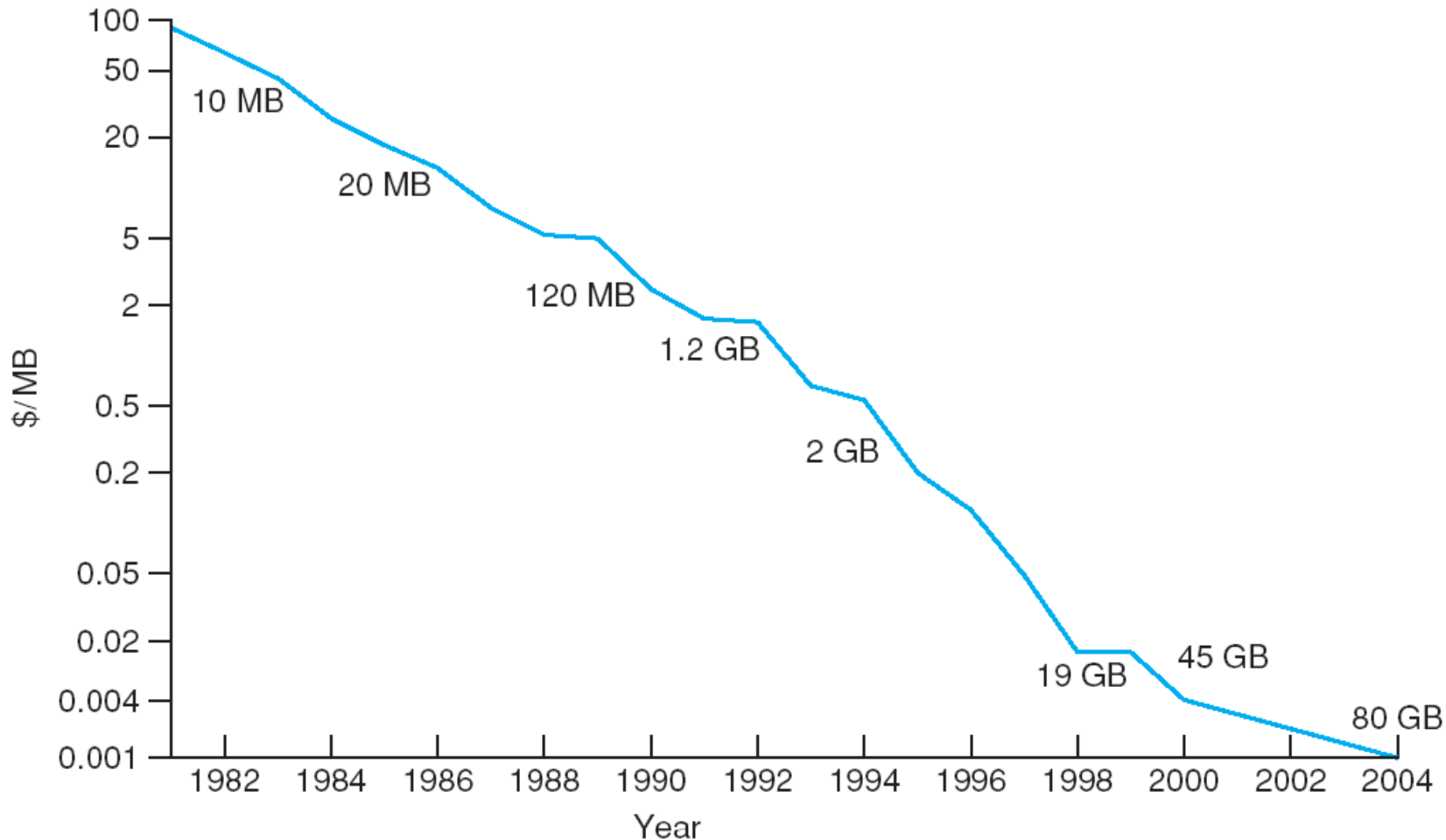
API

- Většina OS pracuje s vyměnitelnými disky (např. disketami, ZIP disky) stejně jako s pevnými disky
 - nový svazek je formátován a na disku se generuje prázdný souborový systém
- Pásy
 - jsou prezentované jako „holé“ (*raw*) paměťové médium
 - aplikace na páskách nemají k dispozici souborový systém
 - otevírají celý páskový mechanismus jako zařízení
 - páskové mechanismy se vesměs nesdílejí, jsou dedikované konkrétní aplikaci
 - OS nepodporují na páskách souborové systémy, aplikace musí samy řešit jak používat bloky dat
 - pásku pak obvykle může používat pouze aplikace, pro kterou byla páska vytvářena (protože jako jediná zná strukturu dat na pásce)

Cena MB RAM (1981 – 2004)



Cena MB pevných disků (1981 – 2004)



Cena MB pásek (1981 -2004)

