Operativni sistemi 2 - Predavanja 1

Upravljanje ulazno/izlaznim uređajima i raspoređivanje diska

Miloš Ivanović

Institut za matematiku i informatiku
PMF Kragujevac

školska 2016/2017. godina

O čemu će biti reči?

O predmetu

- Nastavnik: Miloš Ivanović, asistent: Marko Knežević
- 36 kolokvijumi + 4 prisustvo + 30 seminarski rad + 30 usmeni ispit= 100 bodova
- Uslov za izlazak na završni ispit je propisan Pravilnikom o ocenjivanju, dakle 21 bod na predispitnim obavezama + 35 ukupno sa seminarskim radom
- Dodatni uslov je da se na usmenom delu ispita mora ostvariti bar 10 bodova
- Osnovna literatura: William Stallings, Operating Systems: Internals and Design Principles
- Postoji srpski prevod

Projektovanje ulaza/izlaza

- Verovatno najzahtevniji aspekt projektovanja OS-a
- Složenost prisutna najviše zbog raznorodnosti U/I uređaja
- Uglavnom ćemo se fokusirati na U/I magnetnog diska, jer je to uređaj od koga u najvećem delu zavise performanse savremenog računara

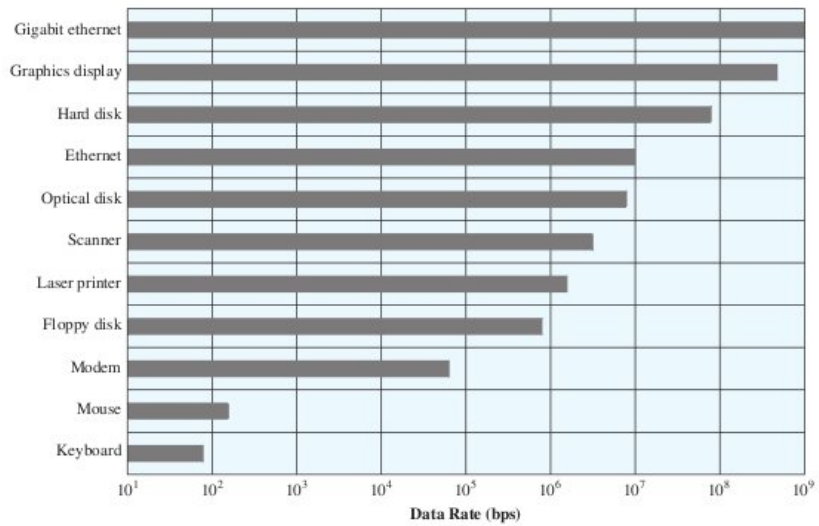
U/I uređaji

Kategorizacija

1. Čitljivi za čoveka - štampači, video terminali, ...
2. Čitljivi za mašinu - kontroleri, diskovi, ...
3. Komunikacioni - ADSL modemi, *WiFi* adapteri, ...

U/I uređaji
Karakteristike

1. Brzina prenosa podataka (štampač nasuprot disku)
2. Primena
3. Složenost upravljanja (štampač-jednostavno, disk-složenije)
4. Jedinica prenosa (blokovi/tokovi)
5. Način predstavljanja podataka (razne vrste kodiranja)
6. Uslovi nastanka i obrada grešaka



Organizacija U/I funkcionalnosti

Tri tehnike za izvođenje U/I operacija

1. Programirani U/I - CPU u ime procesa izdaje komande U/I modulu, a proces čeka dok se operacija ne završi
2. U/I koji se inicira prekidom - CPU izdaje U/I komandu, nastavlja da izvršava naredne instrukcije, a U/I modul ga prekida kada završi zadatu operaciju
3. Direktan pristup memoriji (DMA) - DMA modul upravlja razmenom podataka između U/I i glavne memorije. CPU šalje zahtev za prenos, DMA obavi prenos i prekida CPU kada završi

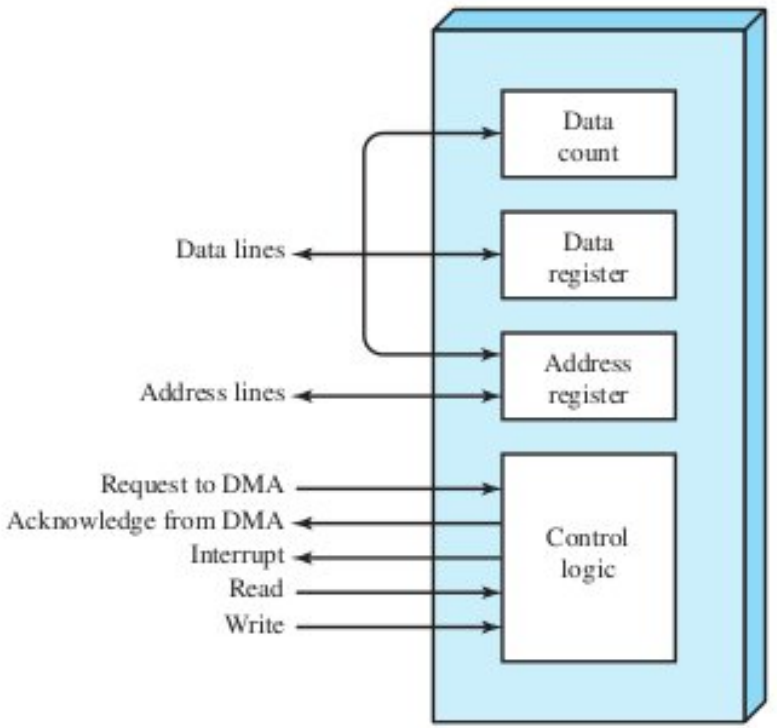
Istorijski razvoj U/I funkcionalnosti

1. **CPU direktno upravlja periferijskim uređajem**
2. **Dodaje se kontroler ili U/I modul**. CPU koristi tehniku programiranog U/I bez *interrupt*-a
3. **Dodaje se kontroler ili U/I modul**. CPU koristi tehniku *interrupt*-a
4. **U/I modul direktno upravlja memorijom preko DMA**
5. **U/I modul postaje poseban procesor**. Sada CPU zadaje čitav niz aktivnosti i biva prekinut samo kada se sve završe.
6. **U/I modulu se dodaje i memorija**, tako da je sada sam po sebi računar. Kontrola terminala.

Kako radi DMA?

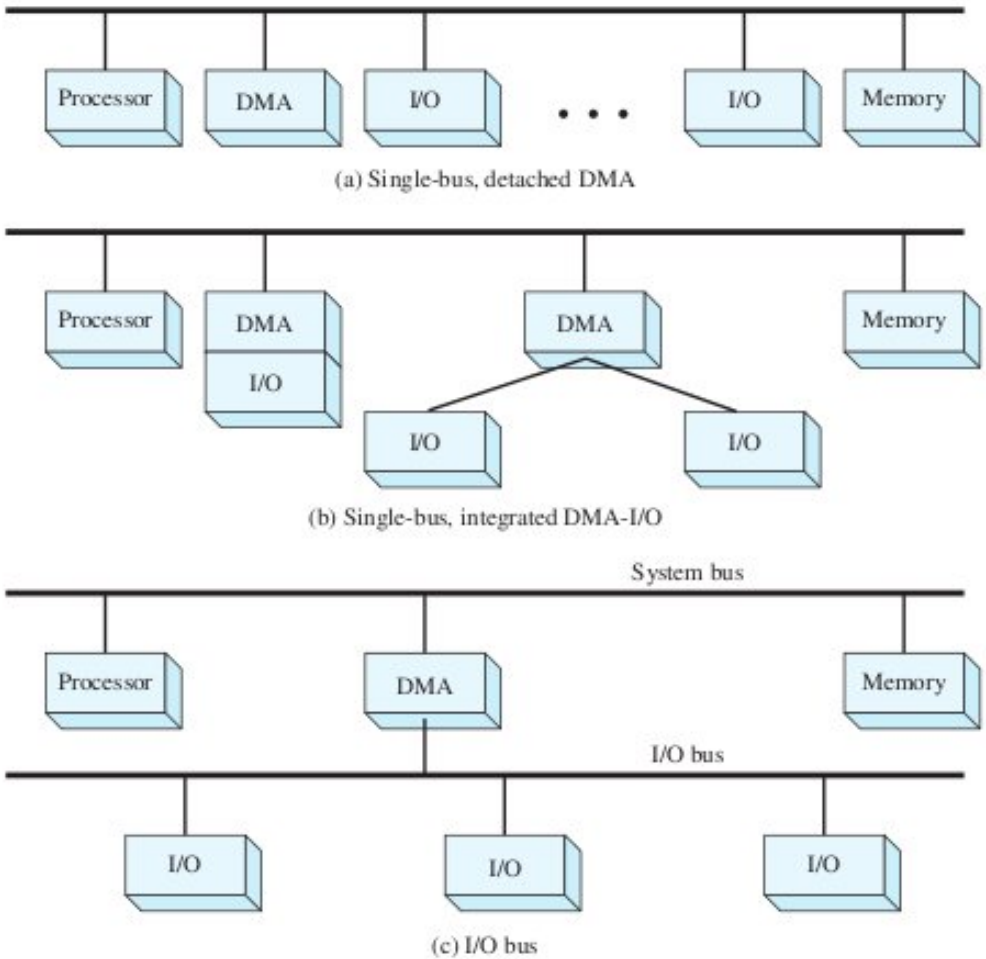
Koje podatke šalje CPU DMA kontroleru?

- Da li se zahteva čitanje/upis?
- Adresa U/I uređaja
- Početnu adresu u memoriji sa koje se čita/na koju se upisuje
- Dužinu podataka za upis/čitanje



Kako radi DMA?

Razni načini hardverske izvedbe



Ciljevi projekta U/I modula OS-a

1. Efikasnost - Kako premostiti ogroman jaz između brzine CPU-a/magistrale i U/I periferija?
2. Opštost - Hijerarhijski, modularni pristup projektovanju U/I funkcije OS-a

Slojevita arhitektura

Svaki sloj izvodi podskup funkcija, tj. obezbeđuje višem sloju odgovarajući interfejs. Niži novoi rade na kraćoj vremenskoj razmeri od viših. Niži nivoi komuniciraju sa hardverom, a viši sa korisnikom.

Slojevita arhitektura U/I funkcije

Lokalni periferijski uređaj - jednostavno

1. Logički U/I - Dozvoljava rad sa datim uređajem preko ID-a, jednostavnih komandi za otvaranje/zatvaranje, upis/čitanje
2. U/I uređaja - Zadate operacije se pretvaraju u sekvence U/I instrukcija, komandi kontrolera itd.
3. Raspoređivanje i upravljanje - Stvarno čekanje u redovima i raspoređivanje hardverskog nivoa.

Primer

http://en.wikipedia.org/wiki/RS-232
http://linux.101hacks.com/unix/stty/

Slojevita arhitektura U/I funkcije

Fajl sistem - složenije

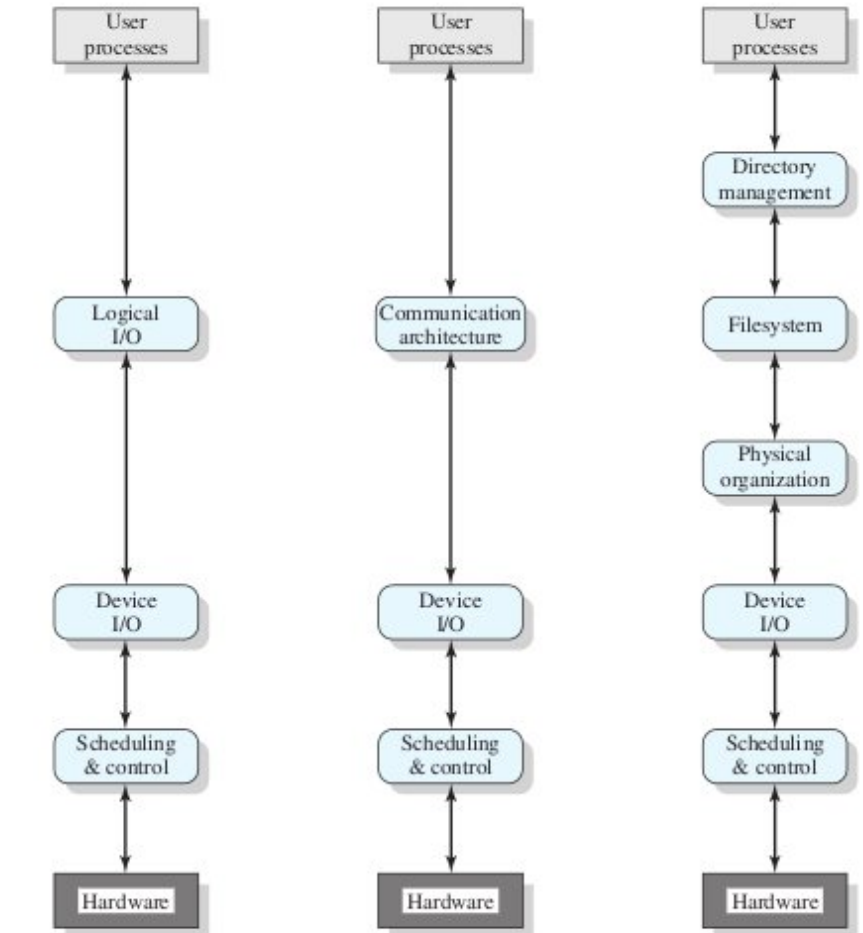
1. Directory management - Simbolička imena fajova se pretvaraju u identifikatore koji referenciraju fajl deskriptore ili tabelu indeksa
2. Fajl sistem - Otvaranje/zatvaranje, čitanje/upis, prava pristupa
3. Fizička organizacija - logičke reference fajlova se pretvaraju u fizičke adrese diska (*track/sector*...)

Primer

http://linux-commands-examples.com/dumpe2fs

Slojevita arhitektura U/I funkcije

Poređenje lokalnog periferijskog uređaja, kom. porta i fajl sistema



(a) Local peripheral device (b) Communications port (c) Filesystem

Baferovanje
Problemi

Primer

Pretpostavimo da korisnički proces čita blokove podataka sa trake, blok=512B. Podaci treba da se učitaju u oblast memorije dodeljene procesu, npr. od 1000-1511. Najjednostavniji način je da se uradi nešto kao `ReadBlock(1000,traka)` i da se čeka na podatke dok postanu raspoloživi (povremena provera ili čekanje na *interrupt*).

Ovakvim pristupom nailazi se na sledeće probleme:

1. Program se zaustavlja čekajući da se završi spori U/I,
2. Ovakav U/I pristup i odluke o *swap*-ovanju se međusobno ometaju jer proces ne može da se prebaci na disk ako čeka U/I.

Baferovanje
Rešenja

Definicija baferovanja

1. Izvođenje ulaznih prenosa pre nego što se zahtevi postave,
2. Izvođenje izlaznih prenosa neko vreme posle postavljanja zahteva.

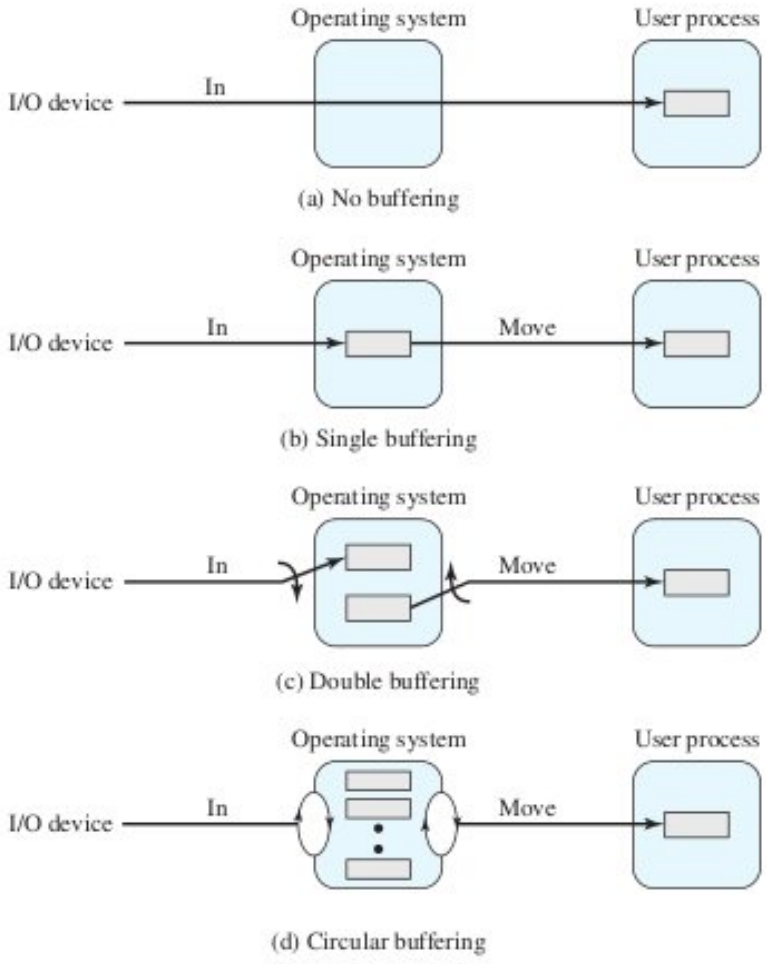
i

Vrste periferala

- Uređaji orijentisani na blokove u baferu skladište ceo blok koji se referiše preko svog rednog broja
- Uređaji orijentisani na tokove u baferu skladište posebne bajtove

Vrste ulaznih bafera

Bez bafera, jednostruki, dvostruki i kružni bafer



Jednostruki bafer

- Čitanje unapred, odn. predviđeni ulaz
- Samo na kraju obrade se bez potrebe učitava blok za većinu zadataka
- OS sada može da *swap*-uje proces jer se bafer nalazi u sistemskoj memoriji, a ne u memoriji sâmog procesa!
- Donekle se usložnjava OS-ova logika koji mora da vodi računa o dodeljivanju sistemskih bafera procesima.

Knut - grubo poređenje performansi

Neka je T vreme potrebno za prenos jednog bloka, a C vreme računanja između U/I zahteva. Bez baferovanja, vreme izvršenja po bloku je $T + C$, a sa jednostrukim baferom oko $max(C, T) + M$, gde je M vreme prenosa iz sistemskog bafera u memoriju procesa.

Dvostruki bafer
Razmena bafera

1. Operaciji se dodeljuju dva sistemska bafera
2. Proces prenosi podatke u jedan bafer (ili iz njega), a operativni sistem prazni (puni) drugi bafer
3. Prenos bloka se grubo može proceniti kao $max(C, T)$
4. Zato blokovski uređaj može da radi punom brzinom ako je $C \leq T$
5. Ako je pak $C > T$, obezbeđuje se da proces ne čeka na U/I
6. Prati se model *producer/consumer*

Kružni bafer

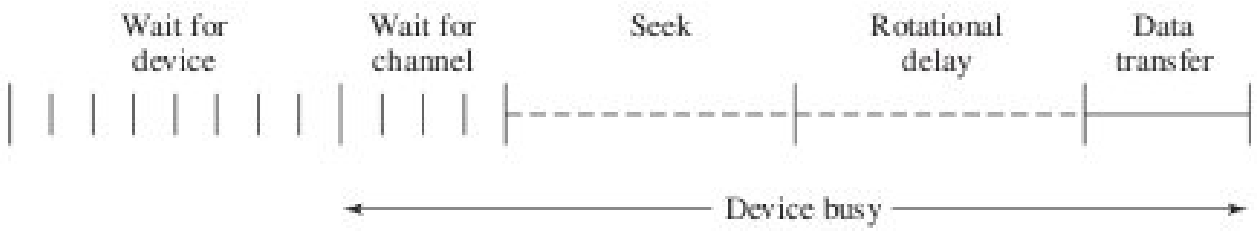
1. Dvostruko baferovanje može da podbaci ako proces izvodi brze neprekidne nizove U/I operacija
2. Zbirka bafera zove se kružni bafer
3. Model *producer/consumer* sa ograničenim baferom

Primer

Komande `vmstat -S M`, `free`, (buffers (FIFO) & cache (LRU))

Optimizacija performansi diska - raspoređivanje

- Disk podsistem je danas za oko četiri reda veličine sporiji od glavne memorije
- Performanse disk podsistema su od ogromnog značaja za ukupne performanse
- Primer: `hdparm -I /dev/sda`



Parametri disk performansi

1. Vreme pozicioniranja - vreme potrebno da se glava diska pomeri na odgovarajuću stazu. Nije linearna funkcija broja staza, već uključuje tzv. smirivanje. Danas obično ispod 10 ms.
2. Rotaciono kašnjenje - vreme potrebno da se disk obrne do odgovarajućeg sektora. Danas $3600 - 15000 rpm$. Za disk od $15000 rpm$ jedan obrtaj $4\,ms$, znači u proseku je kašnjenje $2\,ms$.
3. Vreme prenosa - zavisi od brzine obrtanja diska kao:

$$T = \frac{b}{rN},$$

gde su T - vreme prenosa, b - broj bajtova za prenos, N - broj bajtova po stazi, r -brzina obrtanja u rpm.

Parametri disk performansi

Vremensko poređenje

Ukupno vreme pristupa

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

gde je T_s prosečno vreme pozicioniranja.

i

Primer

Disk sa $T_s = 4\,ms$, $r = 7500\,rpm$, sektori od po $512\,B$, 500 sektora po stazi. Želi se čitanje fajla od 2500 sektora, dakle 5 staza. Vreme za čitanje prve staze je $T_a = 4\,ms + 4\,ms + 8\,ms = 16\,ms$. Ako su staze susedne, onda se svaka sledeća čita za $4 + 8 = 12\,ms$, ukupno $16 + 4 \cdot 12 = 64\,ms$. Ako su pak sektori razbacani po disku, onda imamo za svaki sektor $4\,ms + 4\,ms + 0.016\,ms = 8.016\,ms$, tj. za 500 sektora $4.008\,s$!

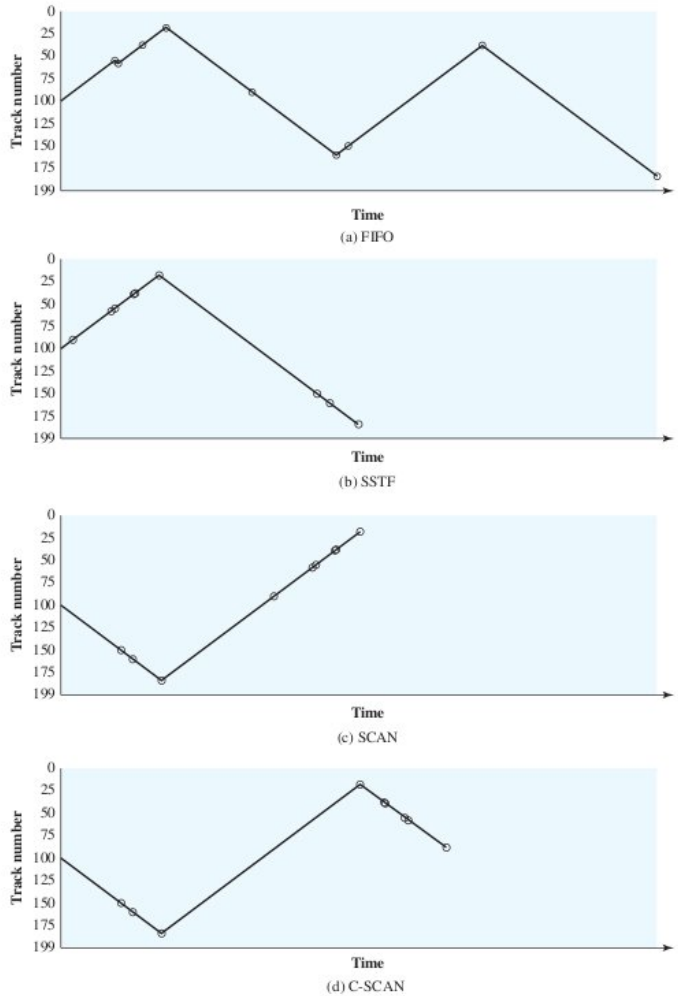
Politike raspoređivanja diska

- Da bi se poboljšale performanse, treba smanjiti srednje vreme koje se provodi u pozicioniranju
- Operativni sistem održava red čekanja zahteva za čitanje/upis od različitih procesa
- Slučajan redosled ispunjavanja zahteva daje veoma loše performanse

Primer

Pretpostavimo da se glava diska na početku nalazi na stazi 100. Zahtevane staze su, redom: 55,58,39,18,90,160,150,38,184. Razmatraju se različite politike raspoređivanja.

Poređenje algoritama za raspoređivanje diska



(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

Algoritmi za raspoređivanje diska

1. FIFO - Najjednostavniji oblik raspoređivanja. Ta strategija je fer, poštuje svaki zahtev, nema izgladnjivanja. Performanse su prihvatljive kada radi jedan proces. Ako ima više procesa - loše performanse.
2. PRI (prioritetno) - Cilj nije da se optimizuje korišćenje diska, već da se ispuni neki drugi zahtev OS-a. Obično se prednost daje kratkim i interaktivnim poslovima.
3. LIFO - Uzima da najskoriji zahtev ima izvesnu prednost. Cilj je poboljšanje lokalnosti, uzevši u obzir da najskoriji zahtev obično sasvim malo pomera glavu diska. Očigledna mogućnost "gladovanja".

Algoritmi za raspoređivanje diska
Nastavak

1. SSTF (Shortest Service Time First) - Politika da se izabere takav zahtev koji zahteva najmanje pomeranje glave. Pošto glava može da ide u oba smera, kada su rastojanja podjednaka, koristi se slučajan izbor.
2. SCAN - Osim FIFO, sve dosadašnje politike mogu neki zahtev da ostave neispunjen, osim ako se red ne isprazni. SCAN zahteva da se glava pomera u samo jednom smeru, usput ispunjavajući zahteve, sve dok ne stigne do poslednje staze ili više nema zahteva u tom smeru. Zatim se obrće. Prednost se daje krajnjim stazama i poslovima koji su najkasnije stigli.

Algoritmi za raspoređivanje diska
Nastavak

1. C-SCAN - Ova politika ograničava skeniranje na samo jedan smer.
2. N-stepeni SCAN - Kod SCAN i C-SCAN moguće je da se glava ne pomera neki duži vremesnki period ako proces često pristupa jednoj stazi. N-stepeni SCAN deli red zahteva na podredove dužine N. Podredovi se obrađuju jedan po jedan koristeći SCAN. Dok se jedan red obrađuje, novi zahtevi stižu u neki drugi red.
3. FSCAN - Slično kao N-stepeni SCAN, koristi dva reda čekanja.

RAID standard

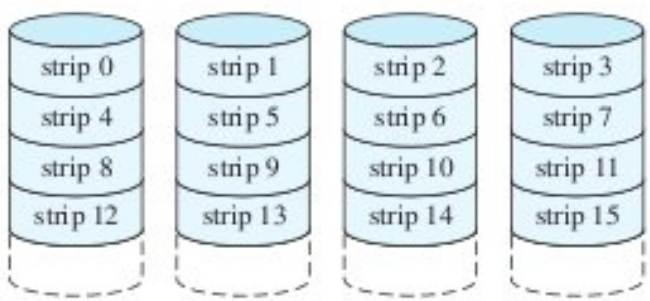
- Dodatni dobici u performansama mogu se dobiti korišćenjem višestrukih paralelnih komponenata
- Sigurnost podataka je takođe bitan faktor
- RAID - *Redundant Array of Independent Disks*

Zajedničke karakteristike RAID nivoa

1. RAID je skup fizičkih uređaja koje sistem vidi kao jedan logički uređaj
2. Podaci su rasporedljeni preko niza fizičkih uređaja
3. Kapacitet redundantnog diska se koristi za skladištenje informacije o parnosti, što garantuje obnovljivost u slučaju otkaza jednog diska

RAID level 0 (stripe)

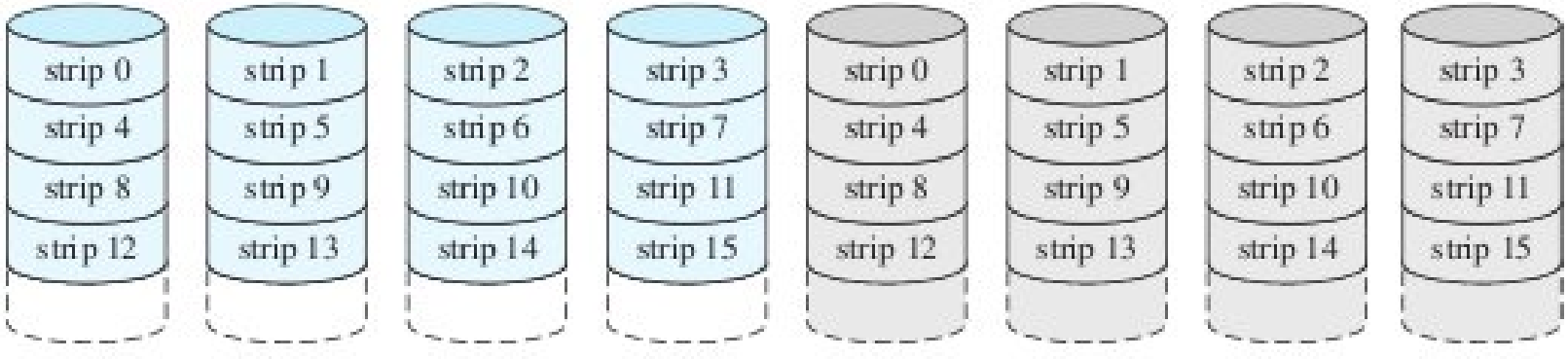
1. RAID0 u stvari nije pravi RAID jer ne uključuje redundansu da bi poboljšao pouzdanost
2. Ako su performanse i kapacitet glavni parametri, RAID0 je dobar izbor
3. Skup logički susednih traka naziva se *stripe*. Recimo, prvi stripe čine prve trake svih n diskova u nizu.
4. Veliki dobici u performansama ako se traže velike količine susednih podataka ili veliki broj konkurentnih zahteva (pitanje velličine *stripe*-a)



(a) RAID 0 (nonredundant)

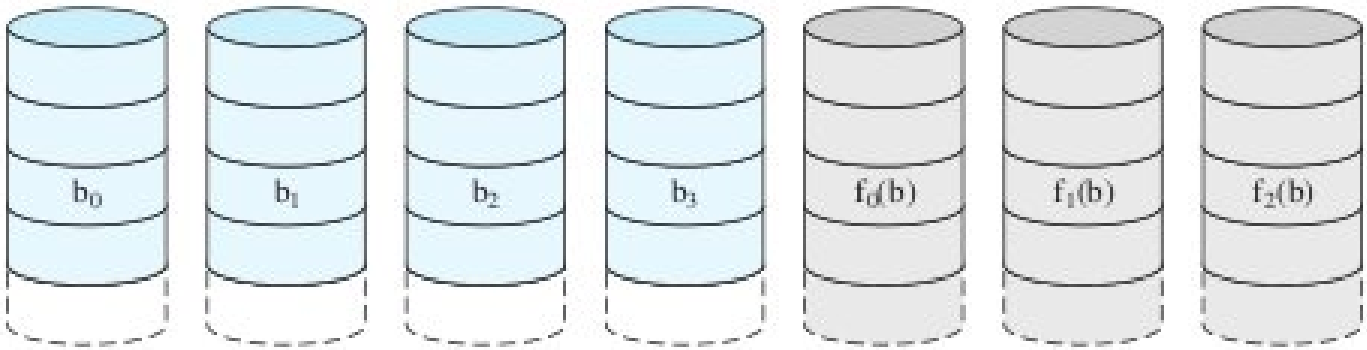
RAID level 1 (*mirror*)

1. Svaka logička traka se preslikava na dva fizička diska
2. Zahtev za čitanje može da se opsluži sa bilo kog od dva diska
3. Zahtev za upisivanje traži da se ažuriraju obe odgovarajuće trake
4. Oporavak posle otkaza je jednostavan
5. Kada ima dosta zahteva za čitanje, približava se performansama RAID0



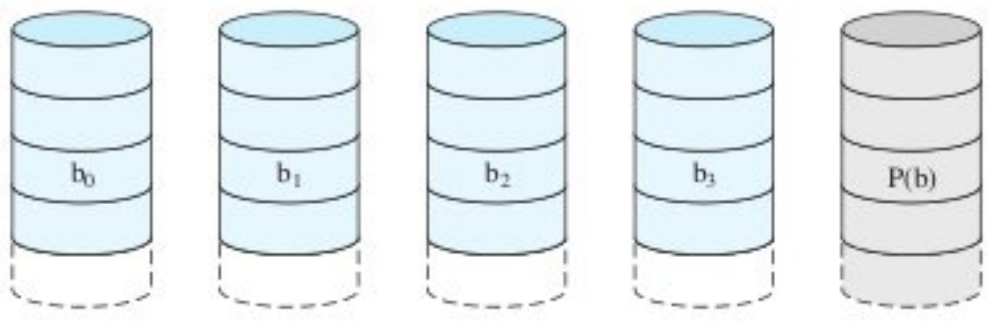
RAID level 2

1. U slučaju RAID2 i RAID3, trake su veoma male, često jedan bajt ili jedna reč
2. Broj kontrolnih diskova proporcionalan logaritmu broja *data* diskova
3. Proračunava se *Hamming*-ov kod za ispravljanje grešaka (detektuje dvostruke i ispravlja jednostruke greške)
4. Prilikom pisanja i čitanja, pristupa se svim diskovima
5. RAID2 ima primenu isključivo kada dolazi do mnogo grešaka diska - skoro nikad



RAID level 3

1. Slično kao RAID2, ali sa samo jednim redundantim diskom
2. Umesto složenog koda za ispravljanje grešaka, koristi se jednostavan bit parnosti za skup pojedinačnih bitova na istim pozicijama na svim diskovima za podatke
3. Ako disk otkaže, pristupa se bitu za parnost i podatak se obnavlja od postojećih
4. Kada se nispravan disk zameni, nedostajući podaci se obnove i nastavlja se s radom



(d) RAID 3 (bit-interleaved parity)

RAID level 3

Otkaz diska i proračun parnosti

Proračun parnosti ako otkaže disk $X1$

$$X_4(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i)$$

Ako se na obe strane jednakosti doda $X_4(i) \oplus X_1(i)$, dobija se:

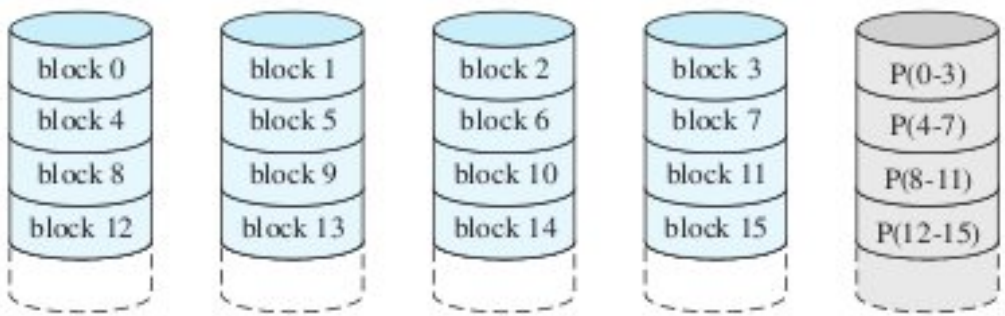
$$X_1(i) = X_4(i) \oplus X_3(i) \oplus X_2(i) \oplus X_0(i)$$

Performanse

1. RAID3 postiže veoma dobre performanse transfera stoga što su podaci podeljeni na veoma male trake duž svih diskova za podatke
2. Na žalost, samo jedan UI zahtev može da se izvršava istovremeno

RAID level 4

1. RAID nivoi 4-6 koriste tehniku nezavisnog pristupa, pa su pogodniji za velike učestanosti U/I zahteva, a manje za velike brzine prenosa
2. Kod RAID 4-6 trake su relativno velike
3. Traka parnosti na disku parnosti se računa bit po bit od odgovarajućih bitova na svakom disku za podatke



(e) RAID 4 (block-level parity)

RAID level 4

Efikasan proračun bita za parnost

Neka je X_4 disk za parnost, a X_1,X_2 i X_3 sadrže podatke. Ako se vrši upisivanje koje obuhvata samo traku na disku:

$$X_4(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i)$$

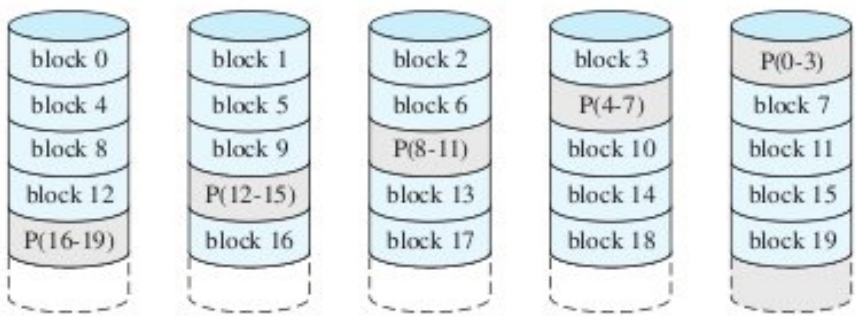
Posle ažuriranja podataka važi:

$$\begin{aligned} X_4'(i) &= X_3(i) \oplus X_2(i) \oplus X_1'(i) \oplus X_0(i) \\ &= X_3(i) \oplus X_2(i) \oplus X_1'(i) \oplus X_0(i) \oplus X_1(i) \oplus X_1(i) \\ &= X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i) \oplus X_1(i) \oplus X_1'(i) \\ &= X_4(i) \oplus X_1(i) \oplus X_1'(i) \end{aligned}$$

Performanse
Svaka operacija upisivanja obuhvata i disk za parnost, pa on može postati usko grlo.

RAID level 5

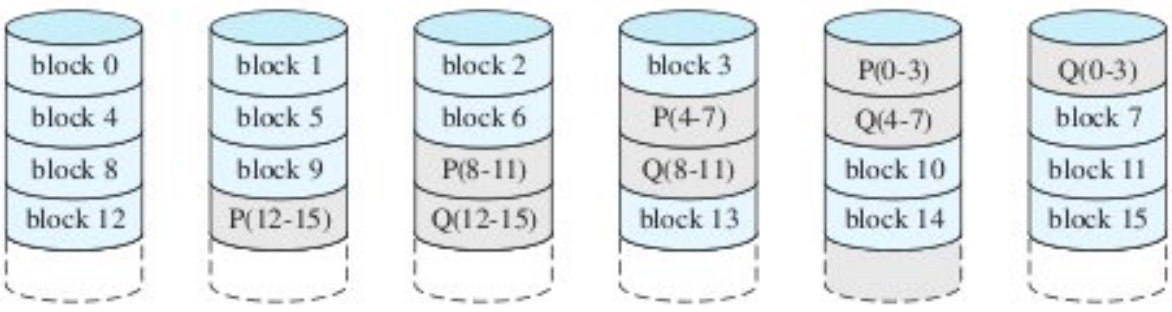
1. Slična organizacija kao RAID4, osim pto se trake za parnost raspoređuju po svim diskovima
2. Koristi se šema kružnog dodeljivanja, kao na slici
3. Za niz od n diskova, traka za parnost je na različitom disku za prvih n traka, a zatim se ponavlja



(f) RAID 5 (block-level distributed parity)

RAID level 6

1. U ovoj šemi se vrše dva odvojena proračuna parnosti i smeštaju i skladište se u posebnim blokovima na različitim diskovima
2. Jedan je XOR, a drugi nezavisan algoritam za proveru podataka
3. Ako korisnik zahteva N diskova za podatke, sistem mora imati $N+2$ diska
4. Podaci mogu da se regenerišu čak i ako otkažu 2 diska
5. Prilično kažnjavanje upisivanja, s obzirom da svako upisivanje utiče na 2 bloka za parnost



(g) RAID 6 (dual redundancy)

Softverski RAID

- Implementiran u sistemima *Windows* od 2000 pa naviše, drajver FTDISK
- Na Linuxu se koristi *MDADM*.
- Može da se primeni nad bilo kojim skupom više diskova, bez obzira na konfiguraciju kontrolera.

Keš diska

- Isti princip kao kod procesorskog keša
- Kada se UI zahtev ne zadovolji iz keša diska, mora da se izbaci jedan blok i novi dovede u keš
- LRU (Least Recently Used) - izbacuje se onaj koji je najduže bio u kešu, a da nije referenciran. Najskorije referenciran na vrhu steka, a izbacuje se sa dna steka
- LFU (Least Frequently Used) - Pridružuje se brojač svakom bloku, koji se svakim referenciranjem poveća za 1. Problem ako se blok retko, ali veliki broj puta referencira

Keš diska

Zamena zasnovana na učestanosti

- Dve sekcije: nova i stara
- U novoj sekciji se brojači ne povećavaju ponovnim referenciranjem
- Izbacuje se samo iz stare sekcije

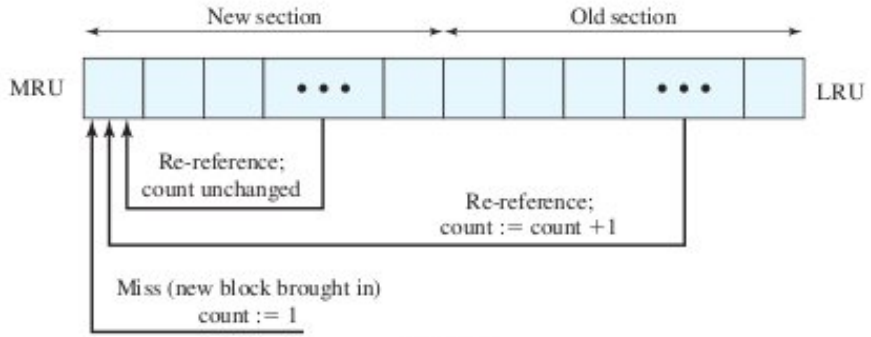


Figure 11.9 **Frequency-Based Replacement**

U/I sistema UNIX

- Svakom pojedinačnom U/I uređaju pridružuje se specijalni fajl (oznake b, c) u direktorijumu /dev
- Postoje dve vrste U/I: baferovani i nebaferovani
- Baferovani U/I prolazi kroz sistemske bafere, a postoje dve vrste bafera:
  1. sistemski bafer keš
  2. karakter red
- Nebaferovani U/I se odvija direktno preko DMA, između U/I modula i procesa

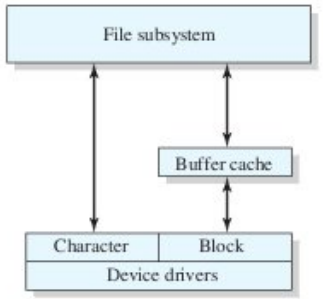


Figure 11.12 UNIX I/O Structure

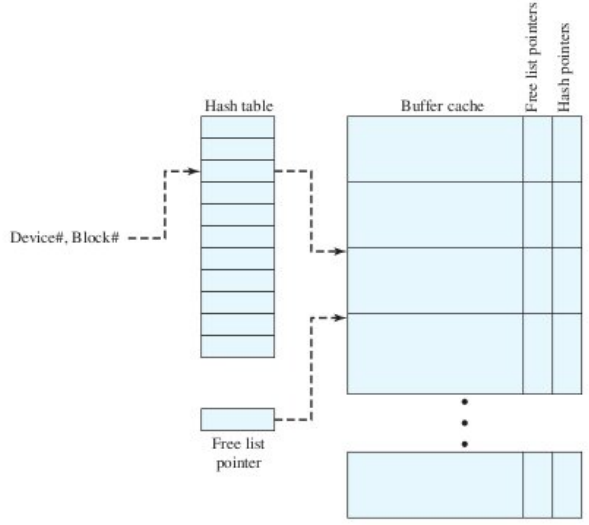
Bafer keš

Bafer keš je u suštini keš diska. Preko DMA kanala se vrši kopiranje iz UI oblasti procesa u bafer keša i obratno.

Održavaju se **tri liste**:

1. Lista slobodnih slotova - svaki slot je veličine sektora na disku
2. Lista uređaja - lista bafera trenutno pridruženih svakom disku
3. U/I red drajvera - lista bafera koji stvarno rade ili čekaju na U/I na određenom uređaju

Za ubrzavanje pretrage koristi se *hash* tabela.



Karakter red

1. U karakter red upisuje UI uređaj, a čita ga proces, ili u njega upisuje proces, a čita ga karakter uređaj
2. U oba slučaja koristi se model producer/consumer
3. Kada se znak pročita, on se u stvari, briše
4. To je suprotno od modela čitalaca/pisaca koji koristi bafer keš

Nebaferovani U/I

1. Mana: Smanjuje mogućnosti za straničenje, tako umanjujući ukupne performanse
2. Mana: UI uređaj vezan s procesom pa je nedostupan drugim procesima

Table 11.5 Device I/O in UNIX

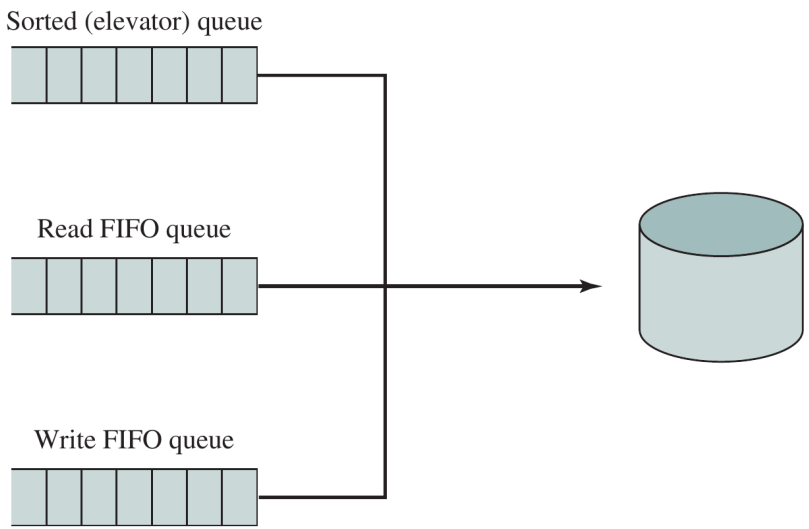
	Unbuffered I/O	Buffer Cache	Character Queue
Disk drive	X	X	
Tape drive	X	X	
Terminals			X
Communication lines			X
Printers	X		X

U/I sistema Linux

- **Linus Elevator** održava jedan red čekanja (i čitanje i upis) i u njemu drži sortirane brojeve bloka uz određene modifikacije (spajanje istih zahteva, strenja zahteva idr.)
- Problem može da bude ako npr. za listu zahteva (10,23,33,41,1001) imamo neprestani dotok zahteva sa niskim brojem bloka. Takođe problem može da bude i isti prioritet čitanja i upisa. Čitanje bi trebalo da bude većeg prioriteta.
- **Raspoređivač po roku** napravljen je kao odgovor na ove probleme. Svaki zahtev ima vreme isticanja, za čitanje podrazumevano 0.5s, a za upis 5s. Svaki zahtev se postavlja u Elevator red i jedan od FIFO redova za čitanje ili upis.
- Najnoviji razvoj obuhvata **Raspoređivač sa predviđanjem** koji je nadređen Raspoređivaču po roku. Kada se pošalje novi zahtev, sistem za raspoređivanje se zadržava $6ms$ jer postoje dobre šanse da stigne još zahteva iste aplikacije za istom oblašću na disku (povećanje lokalnosti).

Raspoređivač po roku (Linux)

I/O raspoređivač i kernel	Test1	Test2
Linus elevator na 2.4	45s	30min
Raspoređivač po roku na 2.6	40s	3.5min
Raspoređivač sa predviđanjem na 2.6	4.6s	15s



Novotarije

- Paralelni fajl sistemi koji se ugrađuju u kernel (Lustre)
- Distribuirani fajl sistemi u *userspace*-u (HDFS)