OS 2. ciklus, prof. Jakobović 2013./2014.

Sadržaj

[6 Monitori 2](#_Toc379019734)

[6.1 Problem proizvođača i potrošača 2](#_Toc379019735)

[6.2 Potpuni zastoj 2](#_Toc379019736)

[6.3 Monitor 3](#_Toc379019737)

[7 Analiza vremenskih svojstava računalnog sustava 7](#_Toc379019738)

[7.1 Uvod 7](#_Toc379019739)

[7.2 Turbo uvod u raspodijele 8](#_Toc379019740)

[7.3 Analiza nedeterminističkog sustava 8](#_Toc379019741)

[7.3 Različite skupine poslova 9](#_Toc379019742)

[7.4 Dodjeljivanje procesora 10](#_Toc379019743)

[8 Spremnički prostor 11](#_Toc379019744)

[8.1 Motivacija 11](#_Toc379019745)

[8.2 Magnetski diskovi 11](#_Toc379019746)

[8.3 Dodjeljivanje spremnika 13](#_Toc379019747)

[8.4 Straničenje (paging) 14](#_Toc379019748)

[9 Datotečni sustav 21](#_Toc379019749)

[9.1 Posluživanje zahtjeva za pristup disku 23](#_Toc379019750)

[Formule 25](#_Toc379019751)

# 6 Monitori

## 6.1 Problem proizvođača i potrošača

1. Neograničen spremnik

sinkronizacijski => OSEM (za potrošača)

OSEM( v = broj nepročitanih poruka), poč v = 0

1. Ograničeni spremnik

N pretinaca

SINKRONIZACIJA: 2 OSEMA-a

PROIZVOĐAČ: v = broj praznih pretinaca, poč v = N

POTROŠAČ: v = broj nepročitanih poruka, poč.v = 0

Više dretvi iste vrste: dodatni BSEM za svaku vrstu, Kritični odsječak -> rad sa spremnikom

Proizvođač (i)

while (1):

proizvedi\_poruku

ispitaj\_osem(2)

ispitaj\_bsem(proizvođač)

MS[UL] = P

UL = (UL+1) % N

postavi\_bsem(proizvođač)

postavi\_osem(1)

1. Red poruka

više parova proizvođača i potrošača, svaki koristi zasebni red poruka

zajedničko skladište pretinaca

SINKRONIZACIJA:

1 OSEM za pretince u skladištu (početno N)

još 1 OSEM za red\_poruka (početno 0)

1 BSEM za zaštitu pristupa skladištu i redovima poruka

## 6.2 Potpuni zastoj

**Problem pet filozofa kao i problem pušača cigareta primjeri su kada može doći do potpunog zastoja**

Uvjeti potpunog zastoja:

1. Barem dvije dretve koriste barem dva sredstva
2. Sredstva se koriste međusobno isključivo
3. Dretva sama otpušta sredstvo nakon korištenja
4. Dretva drži zauzeto sredstvo dok čeka dodjelu dodatnog sredstva (ovaj otklanjamo ako želimo izbjeći potpuni zastoj)

## 6.3 Monitor

* problemi kod semafora
* ispituje samo jedan uvjet – zauzetost jednog sredstva
* ispitivanjem semafora sredstvo se automatski zauzima (nije moguće ispitati više uvjeta i zatim zauzeti sredstvo)
* mogućnost nastanka potpunog zastoja

Za svaki pojedinačni razvojni problem -> posebni monitor

Uporaba monitora temelji se na:

1. Monitorske funkcije (M-funkcije) – korisničke funkcije, piše ih programer za rješavanje određenog sinkronizacijskog problema
2. Jezgrine funkcije (J-funckije) za ostvarivanje monitora – J-funckije koje služe za pisanje monitorskih funckija, MI ih pozivamo SAMO unutar M-funkcija

* unutar M-funkcije dretva provjerava neki složen uvjet te zauzima ili otpušta sredstvo
* u nekom trenutku u M-funkciji može se nalaziti samo jedna dretva, vrijedi sam sve M-funkcije nekog problema
* zaštita M-funkcije uz pomoć monitorskog semafora (BSEM)
* svaki složeni uvjet dobiva svoju oznaku
* ako uvjet nije ispunjen dretva se blokira u redu čekanja tog uvjeta
* svaki uvjet ima posebni red čekanja
* neka druga dretva može ispuniti uvjet i time osloboditi dretvu koja čeka u redu tog uvjeta

J-funkcije monitora

* ulazak u MON
* izlazak iz MON
* stavljanje u red čekanja i propuštanje druge dretve u monitor
* oslobađanje druge dretve iz reda čekanja

Struktura podataka monitora

* red čekanja na ulazu MON
* potreban broj redova čekanja na uvjet

Naše J-funkcije

* ući u MON
* izaći iz MON
* (uvrstiti uz red uvjeta (Mn, K-ti uvjet)) -> neizravno
* osloboditi iz reda uvjeta K

Pomoćna funckija

* čekaj u redu uvjeta (Mn, K)
* uvrstiti ured + uđi u monitor

PRIMJER

2 dretve i 2 sredstva

zastavice\_ VAR, DAT(1 sredstvo je slobodno)

Oblik dretvi

zauzmi\_sredstva(); // M fja

koristi();

otpusti\_sredstva(); // M fjsa

zauzmi\_sredstva {

udi u monitor //cekaj BSEM

dok je (VAR == 0 || DAT == 0)

cekaj na uvjet (m,k)

VAR = DAT = 0; //zauzeto

izadi iz monitora //postavi BSEM

}

otpusti sredstva {

udi u monitor

VAR = DAT = 1

oslobodi iz reda uvjeta(k)

izadi iz monitora

}

PROBLEM PET FILOZOFA (IDEJNO RJEŠENJE)

monitor M

5 uvjeta čekanja (0-4)

lijevi = (I +1) % 5

desni = (I +4) % 5

Oblik dretvi

while (1) {

misliti

uzeti stapice (i)

jesti

spustiti stapice(i)

}

uzeti stapice(i) {

udi u monitor

while (nemam oba stapica)

cekaj na uvjet (m,i)

uzmi stapice

izadi iz monitora

}

otpusti sredstva {

udi u monitor

spusti stapice

oslobodi iz reda uvjeta(m, lijevi)

oslobodi iz reda uvjeta(m, desni)

izadi iz monitora

}

Pravila za monitorske funkcije

* monitorske funkcije izvode s u korisničkom načinu rada, ali zaštićene M-sem (na početku se čeka, a na kraju funkcije postavlja M-sem) - **p\_thread\_mutex\_lock/unlock (M)**
* ako je uvjet zadovoljen dretva zauzima sredstva i koristi ih
* ukoliko neki uvjet nije zadovoljen dretva se svrstava u red toga uvjeta i otključava M-sem -
* funkcije čekaj\_u\_redu (**pthread\_cond\_wait(&uvjet,&kljuc)**) i oslobodi\_iz\_reda (**pthread\_cond\_signal/broadcast(&uvjet)**) pozivaju se samo u M-funkciji (u pravilu)
* uvjet se obično ispituje sa DOK JE (while), a ne AKO JE (if)

ZADATAK – Problem starog mosta

Uvjeti: na starom mostu u isto vrijeme max 3 auta koji voze u istom smjeru

Podaci: 2 reda uvjeta {0,1}, broj = 0, smjer\_na\_mostu (SNM)

auto (smjer (0/1))

na most (smjer)

prijedi

siđi(smjer)

na most(smjer){

mutex\_lock(M)

while( smjer != SNM && broj == 0)

cond\_wait (smjer, M)

broj++;

SNM = smjer;

mutex\_lock (M)

}

sidi(smjer){

mutex\_lock(M)

broj - -

if (broj == 0) //ako si zadnji

cond\_broadcast(1-smjer)

else

cond\_signal(smjer)

ZADATAK – PING-PONG

dretve vrste {0,1}

ispisi (vrsta)

Podaci: 2 reda uvjeta {0,1}, red = 0

ispisi (vrsta) {

mutex\_lock(M)

while (red != vrsta)

cond\_wait(vrsta)

red = 1-vrsta

cond\_signal(1-vrsta)

mutex\_unlock(M)

}

Ova je valjda za manipulaciju broja ispisa. Ping izgleda ovako (adekvatno je valjda i za pong):

ispisi(ping){

mutex\_lock(M)

while (red != ping)

cond\_wait(ping,M)

print „ping“

brojac ++

if (brojac == N)

brojac = 0

red = pong

cond\_broadcast(pong,M)

mutex\_unlock(M)

}

# 7 Analiza vremenskih svojstava računalnog sustava

## 7.1 Uvod

Moguća pitanja

* prosječno poslova u sustavu u nekom trenutku
* prosječno trajanje zadržavanja (odziv)
* kakao poboljšati vremenska svojstva
* kako dodjeljivati dretve procesora

Ocjena ponašanja sustava

* matematički model
* simulacijski model
* mjerenje

Jednostavni matematički modela

Sve pripremne dretve (poslovi) su u jednom redu čekanja.

Pretpostavimo za sada: posluživanje je FCFS (fist come first served), FIFO (redom prispjeća)

Osnovni pojmovi

* trenutak dolaska: ts
* trenutak odlaska: tn
* zadržavanje posla u sustavu: T = tn – ts
* Tr – zadržavanje posla u redu
* Tp - zadržavanje posla u poslužitelju
* T = Tr + Tp

Deterministički sustav

poznati trenutci dolaska i trajanja obrade za sve poslove

PRIMJER

deterministički periodni poslovi:

Ulazne vrijednosti

* alpha – broj dolazaka u jedinici vremena
* beta – broj poslova koje poslužitelj može obaviti u jedinici vremena

Izlazne vrijednosti

* ro – faktor iskorištenja poslužitelja – [formula (1)](#f1)
* T (s crtom iznad) – prosječno zadržavanje poslova u sustavu
* n (s crtom iznad) – prosječan broj poslova u sustavu

Littleovo pravilo – [formula (2)](#f2)

Nedeterministički sustav

dolasci i trajanje obrade su slučajne varijable

alpha – prosječan broj dolazaka u jedinici vremena

beta – prosječan broj poslova koje možemo obaviti u jedinici vremena

## 7.2 Turbo uvod u raspodijele

Slučajna varijabla je varijabla koja poprima slučajne vrijednosti.

Podjela (1)

* kontinuirane (mjesto pada kapljica kiše, na koji ćemo dio stepenice stati…)
* diskretne (broj studenata koji polože OS…)

Podjela (2)

* ograničena
* neograničena

O raspodjelama i slučajnim varijablama

* ponašanje slučajne varijable opisuje raspodjela (jednolika, Gaussova…)
* raspodjela je definirana funkcijom gustoće vjerojatnosti
* vjerojatnost da slučajna varijabla poprimi neku vrijednost dobiva se kao površina ispod funckije gustoće vjerojatnosti
* kontinuirana slučajna varijabla ima vjerojatnost samo za interval
* Poissonova raspodjela – za diskretnu
* eksponencijalna raspodjela – za kontinuiranu

## 7.3 Analiza nedeterminističkog sustava

* prosječan broj dolazaka (alpha) – Poissonova raspodjela
  + vjerojatnost dolazaka k poslova u vremenskom intervalu t
* prosječno trajanje obrade (1 / beta) – eksponencijalna raspodjela
  + vjerojatnost trajanja u nekom intervalu

ZADATAK

Bacamo kocku 10 puta. Kolika je vjerojatnost da ćemo šesticu dobiti 3 puta?

n = 10, k = 3, p = 1/6

(10 povrh 3)

ZADATAK fali, više njih

## 7.3 Različite skupine poslova

PRIMJERI i ZADATCI fale

## 7.4 Dodjeljivanje procesora

Do sada FIFO, sada uvodimo:

Kružno posluživanje (round-robin)

* dretve se obrađuju u kvantu vremena Tq
* nakon jednog kvanta dretva se stavlja na kraj reda pripravnih dretvi
* NEDOSTATAK: više izmjena konteksta

Različiti prioriteti dretvi

1. prioritetnije dretve dobiju više uzastopnih kvanta vremena
2. posebni pripravni redovi za različite prioritete

ZADACI fale

# 8 Spremnički prostor

## 8.1 Motivacija

* izvođenje više procesa istovremeno u zajedničkom spremniku
* kako omogućiti korištenje maksimalnog logičkog adresnog prostora u raspoloživom fizičkom adresnom prostoru
* Danas (zahtjeva i hardwersku podršku):
  + straničenje (paging)
  + virtualna memorija
* struktura pohrane podataka na magnetski diskovima
* najjednostavniji slučaj: 1 proces, u tom trenutku drugi čekaju u pomoćnom spremniku

## 8.2 Magnetski diskovi

Osnovna svojstva magnetskih diskova

NAMJENA:

* pomoćni spremnik prilikom izvođenja procesa
* trajna pohrana podataka (glavna namjena)

Magnetski disk:

1. elektromehanički dio – pohrana podataka
2. upravljački sklop

Adresiranje podataka na disku: redni broj sektora – OS vidi HD kao (0…N) sektora.

Redni broj sektora se preslikava u konkretnu: PLOČA->STAZA->SEKTOR, za što odgovara upravljački sklop.

VREMENSKA SVOJSTVA DISKOVA

ukupno trajanje prijenosa =

trajanje postavljanja glave (head positioning time) =

trajanje traženja staze (seek time) =

ubrzavanje + kretanje max brzinom + usporavanje + fino pozicioniranje

+rotacijsko kašnjenje = Tr / 2

+trajanje prijenosa podataka (data transfer time) =

trajanje čitanja/pisanja dijela ili cijele staze

+trajanje premještanja sa staze na stazu

+rotacijsko kašnjenje

ZADATAK – Vremena

Disk ima 100 sektora/stazi veličine 1 KB, 2000 staza, 1 ploča (površina), brzina vrtnje = 7200 okr/min. Upravljački sklop čita cijelu stazu u interni spremnik. Prijenos u glavni spremnik odvija se 200 Mbt/s, a za to vrijeme ne može se čitat s diska.

1. Kapacitet diska
2. Prosječno trajanje prebacivanja kompaktno smještene datoteke od 135 KB ako je trajanje traženja staze 10 ms i trajanje premještanja sa staze na stazu je 1 ms

IMAMO:  
P = 1

C = 2000

S = 100

M = 1 KB

tseek = 10 ms

tpremj = 1 ms

Odmah možemo dobiti:

Tr = 7200 okr / min => 60/7200 okr /s = 8.33 ms

Rotacijsko kašnjenje (svaki put kad dođemo na novu stazu) = Tr / 2 = 4.17 ms

1. kapacitet = P\*C\*S\*M = 200 000 KB = 200 000 / 1024 MB
2. kompaktno – najbolji mogući slučaj
   1. do početka datoteke

trajanje traženja staze + rotacijsko kašnjenje

tseek + Tr / 2 = 10 ms + 4.17 ms = 14.17 ms

* 1. čitanje cijele staze = Tr = 8.33 ms
  2. prijenos podataka = broj bitova koje premještamo / brzina prijenosa

= (1024 \* 8 \* 100) / (200 \* 10^6) = 4.1 ms

* 1. premještanje sa staze na stazu = 1 ms

između 3 i 4 biramo max vrijednost: MAX(4.1ms, 1 ms) = 4.1 ms

zato što se te dvije operacije odvijaju paralelno ???

* 1. Rotacijsko kašnjenje = Tr / 2 = 4.17 ms
  2. čitaj cijelu stazu = 8.33 ms
  3. prijenos podataka = (1024 \* 8 \* 35) / (200 \* 10^6) = 1.4 ms nema više rotacije pa ju ne moramo pisat (kao što je bio slučaj 3,4)

UKUPNO = 40.5 ms

VARIJANTA: da ima 8 površina? sve bi bilo isto samo bez koraka 4 jer ne bi morali mijenjati stazu već bi čitali sa „paralelne“ staze koja se nalazi na drugoj površini, konačni rezultat bi ostao isti

## 8.3 Dodjeljivanje spremnika

ciljevi:

* smještaj više procesa istodobno
* smještanje procesa koji je veći od raspoloživog (fizičkog) spremnika

evidencija o procesima: za svaki proces – procesni informacijski blok

1. Statičko dodjeljivanje spremnika

* apsolutni oblik programa
  + adrese u programu -> stvarne lokacije u spremniku
  + mora biti poznata početna adresa (PA)
* relativni oblik programa (sadrži relativne adrese)
  + PA = 0
  + prije izvođenja potrebna je pretvorba u apsolutni oblik A.A. = R,A, + PA
* izvođenje više procesa: svaki proces ima različite PA
* za više procesa postoji više particija s različitim PA koje su unaprijed definirane
* programi se pripremaju za određenu particiju
  + zato se zove statičko dodjeljivanje
  + proces se uvije nalazi u istoj particiji
* fragmentacija kod statičkog dodjeljivanja
  + unutarnja fragmentacija – neiskorišteni dio particije
  + vanjska fragmentacija – nema procesa priređenih za slobodnu particiju

1. Dinamičko raspoređivanje
   * IDEJA: dobavljanje PA se obavlja sklopovski
   * program uvijek ostaje u relativnom obliku
   * program koristi LAP
   * LAP -> PA = 0
   * program se smješta na bilo koju PA -> FAP
   * DODACI PROCESORU
     + bazni registar – pamti PA aktivne dretve
     + zbrajalo – pribraja PA na logičku adresu
   * bazni registar postaje dio konteksta dretve (zajednički za isti proces)
   * PROBLEM: nezavisnost procesnog adresnog prosotra
   * sklopovsko rješenje: uvodi se registar ograde koji predstavlja gornju granicu procesnog prostora
   * prilikom narušenja granice generira se prekid
   * DODATNO: posebni bazni registari i ograde za različite segmente (instrukcijski, stogovni, podatkovni)
   * fragmentacija dinamičkog raspoređivanja: nastanak rupa između zauzetih adresnih prostora
   * suzbijanje fragmentacije
     + prilikom oslobođenja rupe se spajaju
     + prilikom dodjele traži se najmanji mogući prostor
   * skupljanje otpadaka (garbage collector): povremeno slaganje procesnog adresnog prostora u kompaktni niz

PRIMJER

Koliki je broj rupa (n) uz zadani broj blokova (m)?

pretpostavka: vjerojatnost oslobađanja = vjerojatnost zahtjeva

n = f(m)

m = a+b+c+d

m = (2b+2d) / 2 = b + d

Vjerojatnost povećanja broja rupa za 1 (Pn+1)

Pn+1 = Posl \* (a/n)

P n-- ???

a/n = d /n => a = d

m = (2d + 2b) = 2 (b+d) = n

n = m / 2

Kruthovo 50 % pravilo – broj rupa = polovica broja punih blokova

1. Prekidni način rada
   * program
     + osnovni dio (uvijek u spremniku)
     + promijenjivi dio (u spremniku po potrebi)

## 8.4 Straničenje (paging)

* relativni (logičke adrese); logički adresni prostor (LAP)
* apsolutne (fizičke adrese): fizički adresni prostor (FAP)
* LAP se dijeli na stranice (pages)
* FAP se dijeli na okvire (frames)
* veličina okvira = veličina stranice

Kako radi straničenje:

1. stranice LAP se automatski pohranjuju u okvire (kada ih proces zatraži) => omoguće postojanje (smještaj) više procesa
2. stranica se može privremeno izbaciti (i pohraniti na disk) => LAP može biti veći od FAP
3. stranica se može staviti u bilo koji okvir => nema fragmentacije
4. kad se stranica više ne koristi okvir postaje slobodan

Za ostvarenje straničenja:

1. za svaku stranicu svakog procesa moramo znati u kojem okviru se nalazi
2. pretvorba LA u FA mora biti riješena sklopovski
3. koju stranicu privremeno izbaciti ako trebamo slobodan okvir

* pretvorba LA u FA
* za svaki proces OS gradi tablicu prevođenja
* u tablici se za svaku korištenu stranicu zapisuje adresa okvira u kojem se stranica nalazi

PRIMJER

LAP = 23 stranica sa po 4 B ( 25 adresa)

FAP = 6 okvira (6 x 4 B)

PROCES

|  |  |
| --- | --- |
| 0 | AAAA |
| 1 | BBBB |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 | CCCC |

FAP

|  |  |
| --- | --- |
| 0 |  |
| 1 | AAAA |
| 2 | CCCC |
| 3 |  |
| 4 | BBBB |
| 5 |  |

npr. 2. slovo B

LA: 001|01 (1. stranica 2. mjesto) r = 3 bita, p = 2 bita

Tablica prevođenja (svaki proces ima svoju)

|  |  |  |
| --- | --- | --- |
| 0 | 001 | 1 |
| 1 | 100 | 1 |
| 2 | … | 0 |
| 3 | … | 0 |
| 4 | … | 0 |
| 5 | … | 0 |
| 6 | … | 0 |
| 7 | 010 | 1 |

gdje je prvi stupac broj retka, a drugi stupac broj okvira (u FAP-u),a treći stupac bit prisutnosti

=> LA: 001|01 -> FA: 100|01 (unutar procesora), 001 ima 3 bita, q = 3

prvi dio (adresa->okvir) čitamo iz tablice prevođenja, a drugi dio samo prepišemo (pomak unutar retka = pomak unutar okvira)

* bit prisutnosti procesa – nalazi li se stranica u radnoj memoriji (zapisan u tablicu prevođenja)
* pokušaj dohvata neprisutne stranice izaziva grešku
* dijelovi LAP moraju se prvo rezervirati
* samo rezervirani dijelovi mogu se dodijeliti

Procesni informacijski blok

* opisnici dretvi
* opisnik virtualnog procesnog adresnog prostora
* ostali podaci: korisnik, izvršna datoteka…

Opisnik virualnog procesnog adresnog prostora

* tablica rezerviranog adresnog prostora (poč-kraj; poč-kraj;…)
* smještaj dodijeljenih stranica na diku
* tablica prevođenja

Višerazinska tablica prevođenja

LAP => 28 stranica, 24 veličina stranice 256 stranica, 16 podataka

Proces koristi jednu 1 stranica

LA => 1111 1110|1010 (redni broj stranice | adresa unutar stranice)

Jednorazinska tablica

|  |  |
| --- | --- |
| 0000 0000 | … |
| … | … |
| 1111 1110 | 14 |
| 1111 1111 | … |

Dvorazinska tablica

|  |  |
| --- | --- |
| 0000 |  |
| … |  |
| 1111 | \* (pointer) |

|  |  |
| --- | --- |
| 0000 |  |
| … |  |
| 1110 | 24 |
| 1111 |  |

* tablica gornje razine –> page directory
* tablica donje razine –> page table
* zbog uštede prostora za smještaj tablice prevođenja

Sklopovska podrška prevođenju LA->FA

Straničenje na zahtjev

* prilikom adresiranja nepostojeće stranice događa se prekid
* nakon ove vrste prekida mora se ponoviti prekinuta instrukcija
* proces sadrži skrivene kopije registara

Strategija zamjene stranica

* bit pristupa (A) – access -> stranica je korištena
* bit nečistoće (D) – stranica je mijenjana -> mora se zapisati na disk prilikom izbacivanja
* bit GL – stranica je globalno djeljiva

Najvažnije teorijske strategije

* FIFO – izbacuje se najstarija stranica
* LRU – izbacuje se stranica koja se najdulje nije koristila
* LFU – izbacuje se stranica koja se najmanje puta koristila
* OPT – izbacuje se stranica koja se najdalje u budućnost neće korisiti

ZADATAK sa straničenjem

proces veličine 400 riječi (1-400) generira slijed:

23,47,333,81,105,1,400,157,30,209,289,149,360

proces ima na raspolaganju 200 riječi u privremenoj memoriji veličine stranice: 50 riječi. Odredite postotak promašaja za FIFO, LRU, LFU i OPT strategije

Odmah računamo:

FAP = 200 / 50 = 4 okvira

LAP = 400 / 50 = 8 stranica

brojeve stranica određujemo temeljem npr. 23 / 50 = 0. nešto, dakle stranica je 1. 333 / 50 = 6. nešto, dakle stranica je 7 itd.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stranice | 1 | 1 | 7 | 2 | 3 | 1 | 8 | 4 | 1 | 5 | 6 | 3 | 8 | Promašaji |
| FIFO | **1** | HIT | 1 | 1 | 1 | HIT | **8** | 8 | 8 | 8 | **6** | 6 | 6 | 11/13 |
| - | **7** | 7 | 7 | 7 | **4** | 4 | 4 | 4 | **3** | 3 |
| - | **-** | **2** | 2 | 2 | 2 | **1** | 1 | 1 | 1 | **8** |
| - | - | - | **3** | 3 | 3 | 3 | **5** | 5 | 5 | 5 |
| LRU | **1** | HIT | 1 | 1 | 1 | HIT | 1 | 1 | HIT | 1 | 1 | 1 | **8** | 10/13 |
| - | **7** | 7 | 7 | **8** | 8 | 8 | **6** | 6 | 6 |
| - | **-** | **2** | 2 | 2 | **4** | 4 | 4 | **3** | 3 |
| - | - | - | **3** | 3 | 3 | **5** | 5 | 5 | 5 |
| LFU | **1** | HIT | 1 | 1 | 1 | HIT | 1 | 1 | HIT | 1 | 1 | HIT | 1 | 9/13 |
| - | **7** | 7 | 7 | **8** | **4** | **5** | **6** | **8** |
| - | **-** | **2** | 2 | 2 | 2 | 2 | 2 | 2 |
| - | - | - | **3** | 3 | 3 | 3 | 3 | 3 |
| OPT | **1** | HIT- | 1 | 1 | 1 | HIT | 1 | 1 | HIT | **5** | **6** | HIT | HIT | 8/13 |
| - | **7** | 7 | 7 | **8** | 8 | 8 | 8 |
| - | **-** | **2** | 2 | 2 | **4** | 4 | 4 |
| - | - | - | **3** | 3 | 3 | 3 | 3 |

Dodatak: prikažite trenutni izlaz tablice prevođenja na karju primjene OPT strategije

|  |  |  |
| --- | --- | --- |
| 1 | (1) | 0 |
| 2 | (3) | 0 |
| 3 | 4 | 1 |
| 4 | 3 | 1 |
| 5 | (1) | 0 |
| 6 | 1 | 1 |
| 7 | (2) | 0 |
| 8 | 2 | 1 |

gdje je prvi stupac broj stranice, drugi broj okvira u kojem se stranica nalazi (ili se nalazila prije nego li ju je netko izbacio), a treći stupac je bit prisutnosti

Praktične izvedbe zamjene stranica

* sklopovska izvedba najčešće koristi samo bit pristupa i bit čistoće
* npr. satni algoritam (second chance, clock algorithm)

ZADATAK sa satnim algoritmom

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CLOCK | 2 | 1 | 2 | 3 | 5 | 3 | 4 | 5 | 6 | 3 | 2 | 6 |
| **2** | 2 | HIT | 2 | 2 | HIT | **4** | HIT | 4 | **3** | 3 | HIT |
| - | **1** | 1 | **5** | 5 | 5 | 5 | **2** |
| - | - | **3** | 3 | 3 | **6** | 6 | 6 |
| BIT  PRISTUPA | **1** | 0 | 1 | 1 | 0 | 0 | **1** | **1** | 0 | **1** | 0 | 0 |
| 0 | **1** | **1** | 0 | **1** | **1** | 0 | 1 | 0 | 0 | **1** | **1** |
| 0 | 0 | 0 | **1** | 0 | 1 | 0 | 0 | **1** | 0 | 0 | 1 |

**bold jedinica** označava gdje smo trenutno (koji bit prisutnosti gledamo)

Raspodjela okvira u višeprogramskom radu

OS vodi evidenciju o korištenju okvirima (opisnik okvira)

Stanja okvira

* aktivno – dodijeljen nekom procesu
* slobodno – trenutno se ne koristi
* obrisan (zeroed) – spreman za dodjeljivanje

Međustanja

* prenošenje stranica na disk
* neispravan okvir

Broj okvira koje se dodjeljuje nekom procesu je početno ograničen.

„Working set“ – skup okvira trenutno dodijeljen nekom procesu.

Struktura podataka jezgre OS-a trajno je smještena u rezerviranim okvirima.

MOŽDA NEŠTO fali???

Programiranje uz straničenje

* API funkcijama moguće je utjecati na raspodjelu okvira i algoritam zamjene stranica
* različite strukture podataka i algoritmi mogu imati velik utjecaj na performanse

ZADATAK sa straničenjem

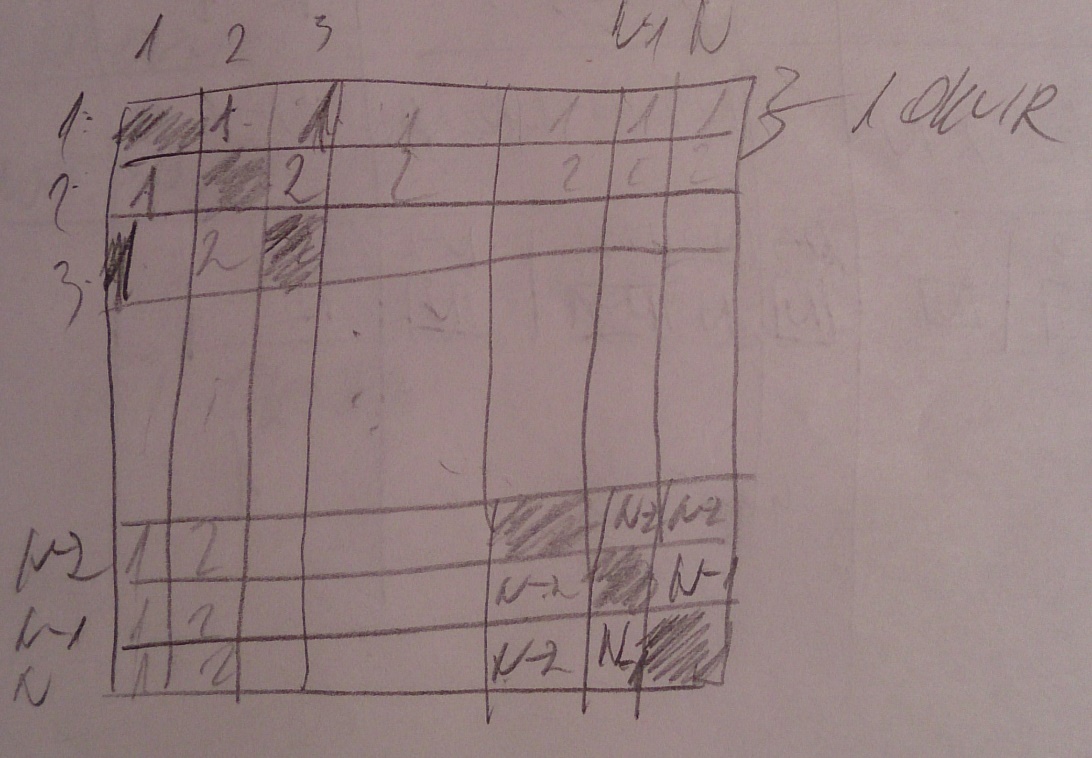
t = 0

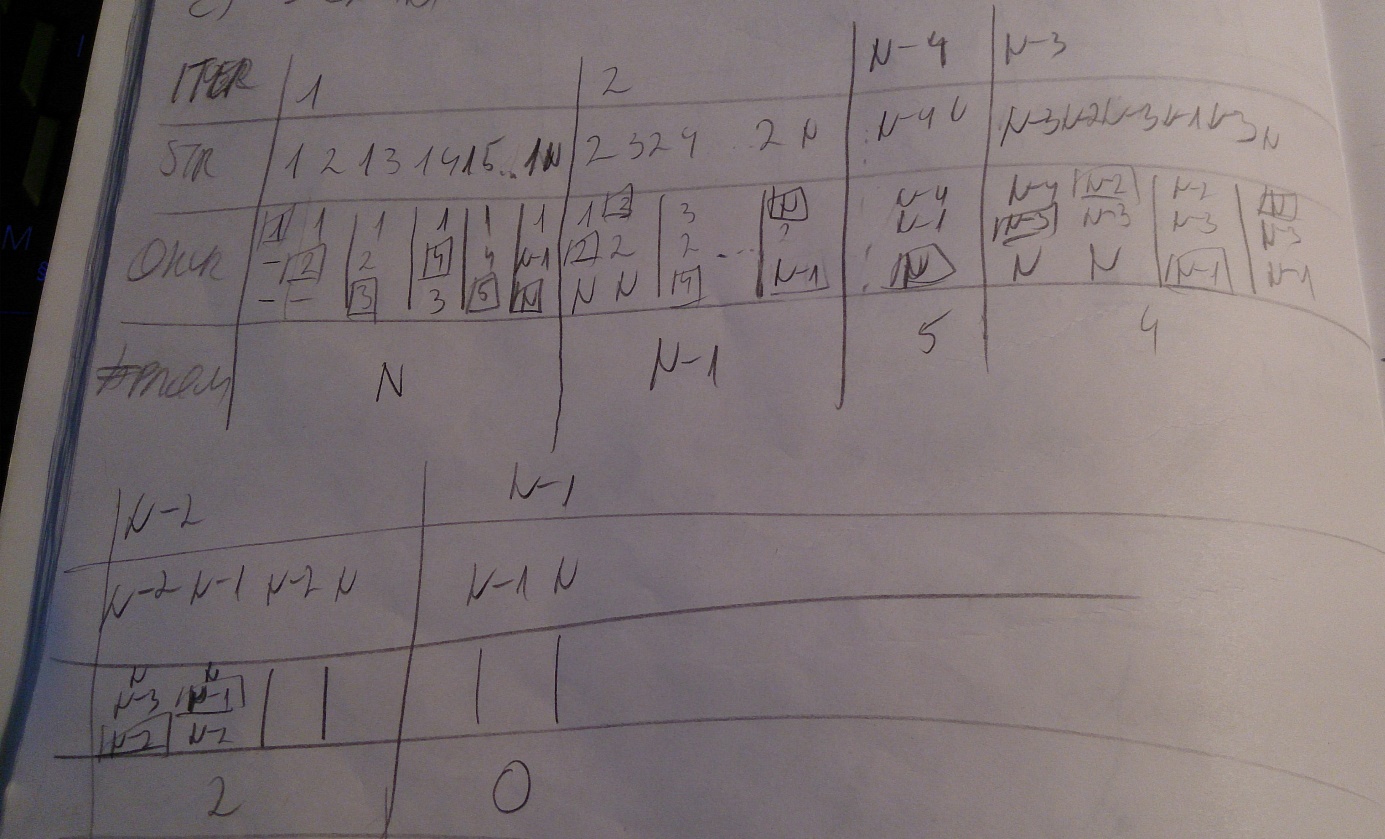
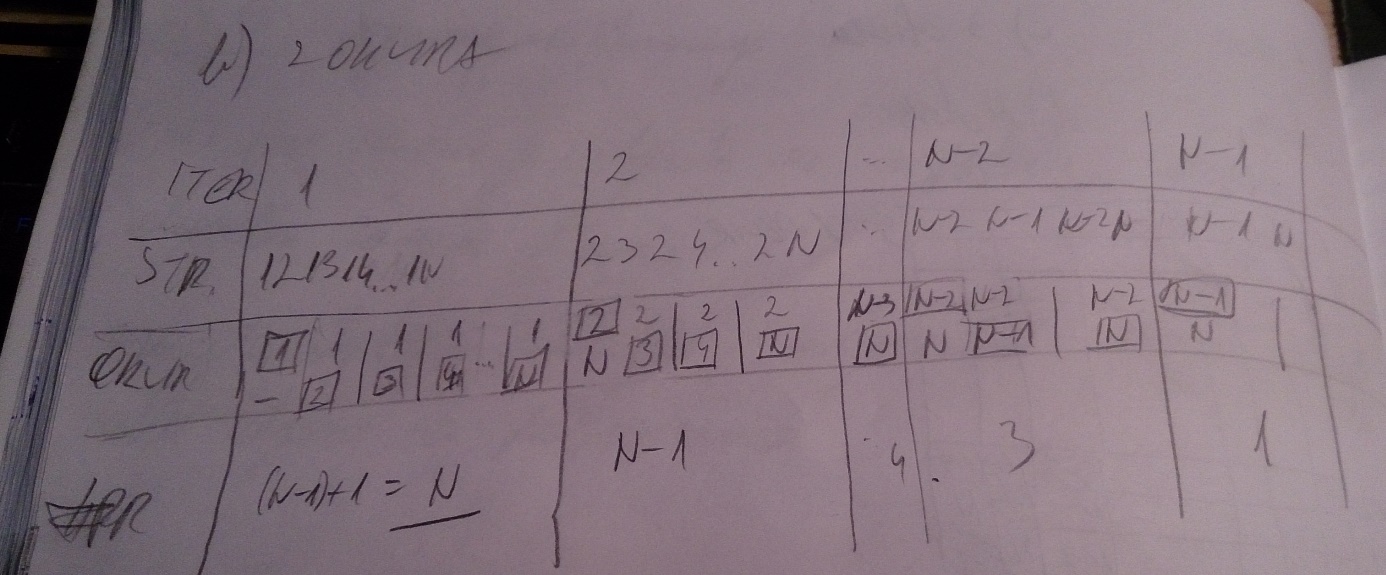
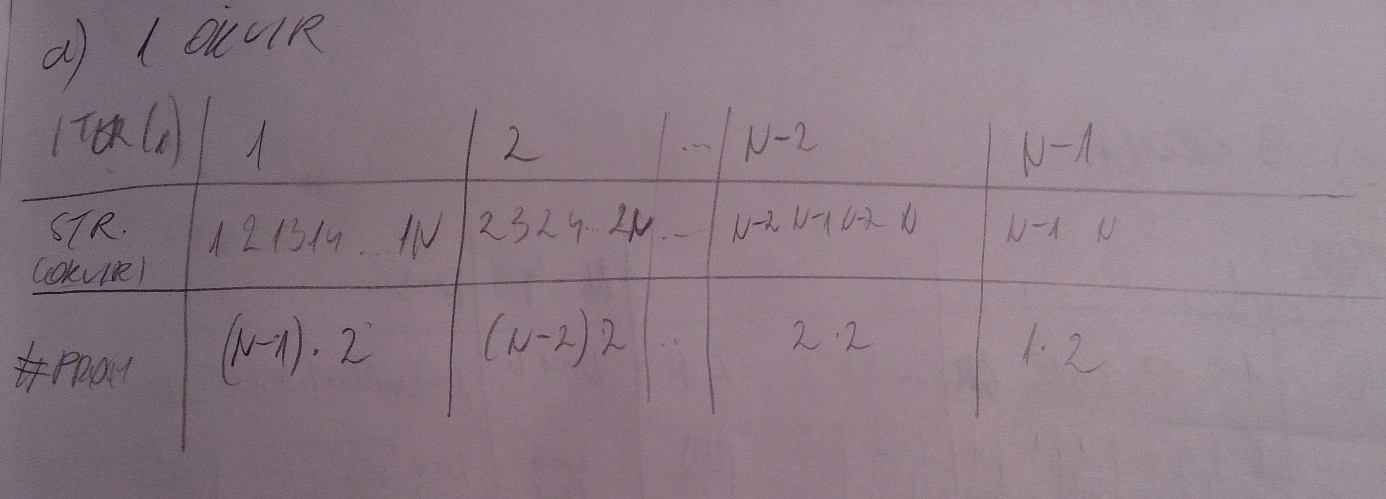
for (i=1 do N-1)

for(j =i+1 do N)

t += A[ i,j]

t\*= A [j,i]

* u sustavu s straničenjem veličina okvira je N riječi, a algoritam zamjejene stranica je LRU
* matrica A[1…N, 1…N] je pohranjena po retcima
* koliko promašaja će izazvati zadani program ako za A u radnom spremniku postoji
  1. 1 okvir
  2. 2 okvira
  3. 3 okvira
  4. N okvira



1. ukupno promašaja = [ (N-1) \* N] /2 \*2 = N^2 – N
2. ukupno promašaja = [N\*(N+1) / 2] – 2
3. ukupno promašaja = [N\*(N+1) / 2] – (1+3)
4. ukupno promašaja = N

# 9 Datotečni sustav

Uloga datoteka

* trajna pohrana podataka
* komunikacija između procesa

Opisnik datoteke

* atributi: naziv, vrsta, vlasnik, prava pristupa, vrijeme stvaranja, vrijeme promjene

Opis smještaja datoteke na disku

DISK 1--------N PARTICIJA 1--------1 DATOTEČNA TABLICA

Datotečna tablica

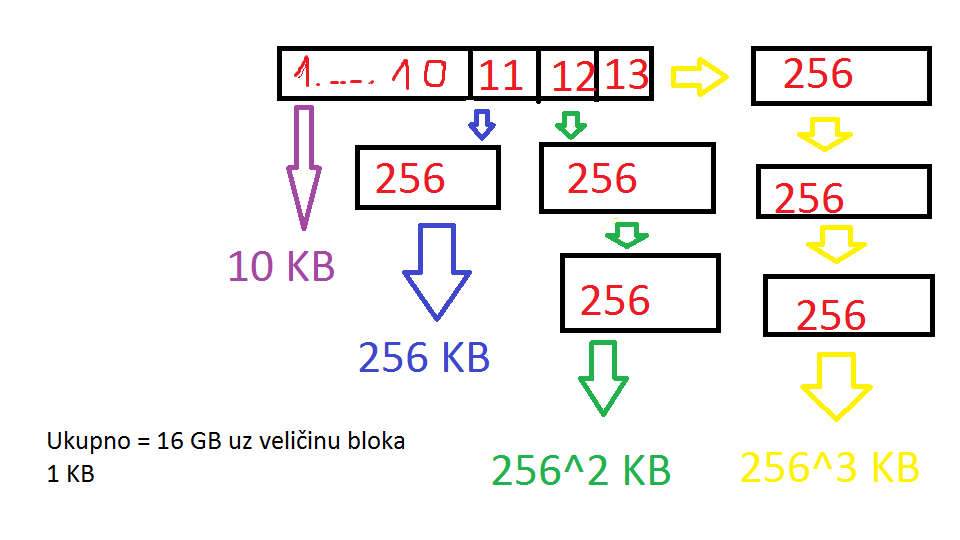
* opisnici datoteka
* opis slobodnog prostora
* realizirano bitovnim prikazom ili listom

Smještaj datoteke – datoteka se dijeli na blokove koji na disku mogu biti zapisani proizvoljnim redom

Opis smještaja – lista pointera/kazaljki na blokove

I-node

|  |  |  |  |
| --- | --- | --- | --- |
| 1….10 | 11 | 12 | 13 |



NTSF

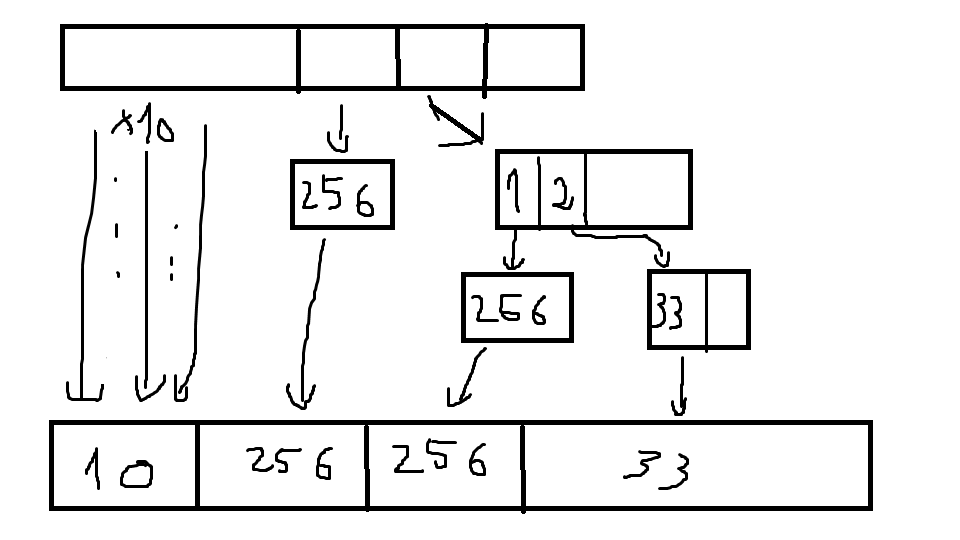
* MFT – master file table – sadrži opisnike datoteka
* LCN – logical cluster number: redni broj bloka na disku (kao fizička adresa)
* VCN – virtual cluster number: redni broj bloka unutar datoteke

Tablica smještaja: VCN|POČETAK LCN|BROJ BLOKOVA

ZADATAK I-node

I-node, pohranjena je datoteka 555 KB; koliko prostora zauzimaju kazaljke za tu datoteku ako je veličina bloka 1 KB, a veličina kazaljke 32 bita.

NOTE: sve veličine su u KB



imamo 4 bloka svaki po 1 KB, dakle rješenje je 4 KB

VARIJACIJA: izračunaj za datoteku od 3 GB, [rješenje](#rjInode)

## 9.1 Posluživanje zahtjeva za pristup disku

određivanje redoslijeda nagomilanih zahtjeva za pristup disku

Ciljevi

1. propusnost (broj posluženih zahtjeva u jedinici vremena)
2. odziv (prosječno čekanje na zahtjev)

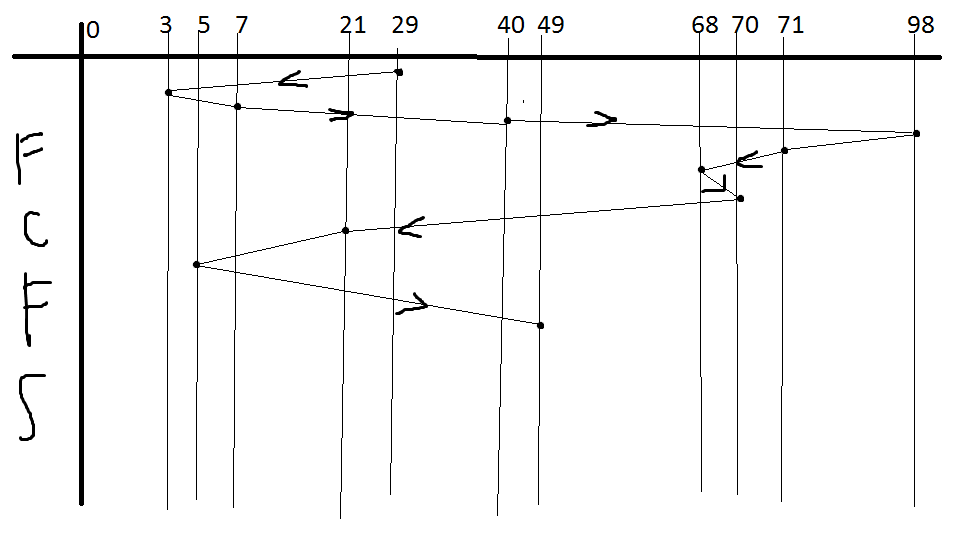
Strategije

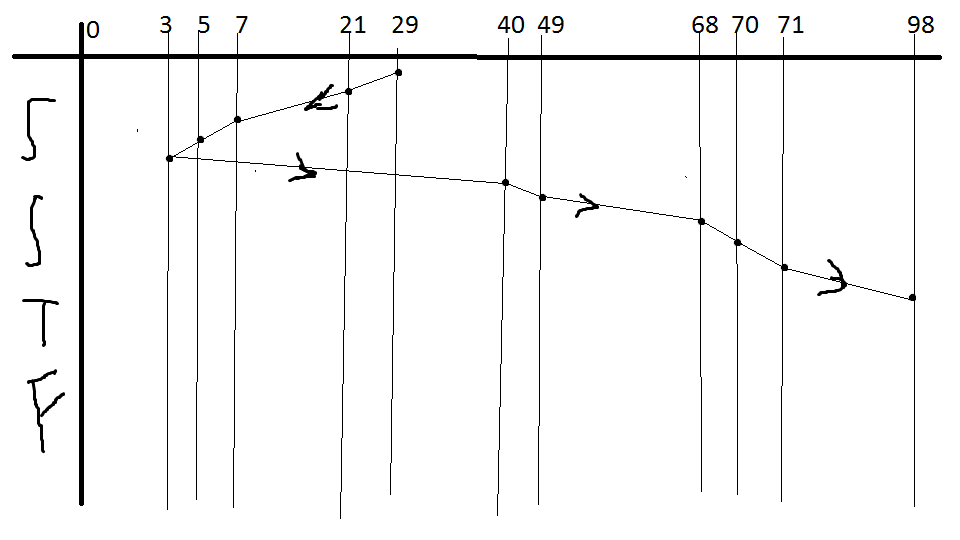
* FCFS (fist come first served) – odnosno FIFO – redom prispjeća
* SSTF (shortest seektime first) – može doći do izgladnjivanja
* LOOK/SCAN (pregledavanje)
* C-LOOK/C-SCAN (kružno pregledavanje)

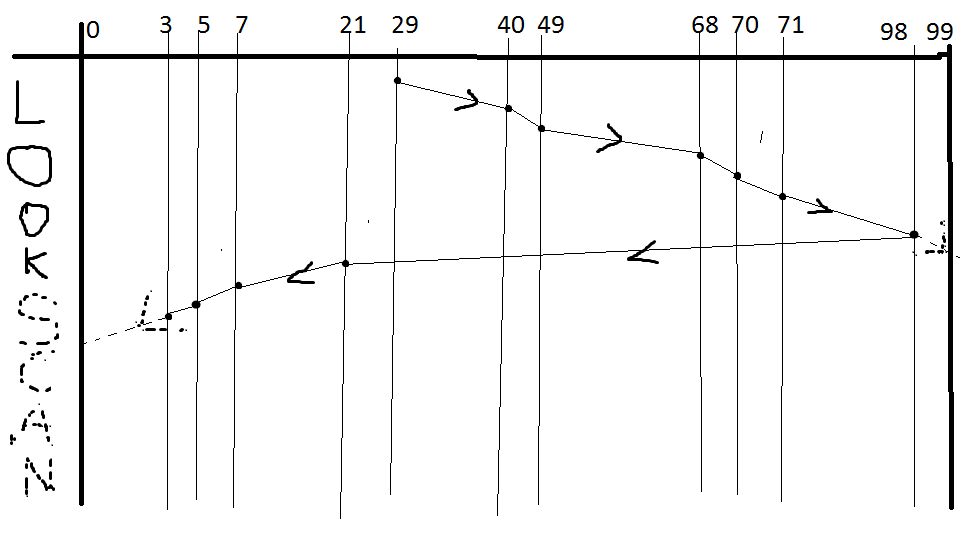
prilikom implementacije strategije mogu uzimati u obzir samo ograničen broj najstarijih zahtjeva

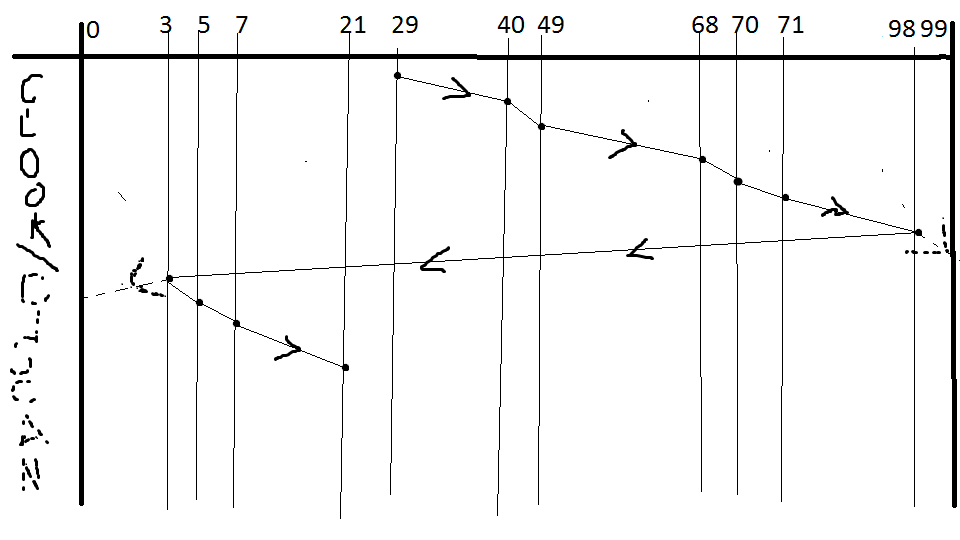
ZADATAK strategije

Disk ima 100 staza (0-99), glava se trenutno nalazi na stazi 29 s tim da je prije bila na stazi 8. Zahtjevi za pristup stazama po redu prispjeća su: 3,7,40,98,71,68,70,21,5,4. Skicirajte kretanje glave prilikom obrade zahtjeva za sve navedene zahtjeve.









Kod LOOK/SCAN odnosno C-LOOK/C-SCAN važno je da je glava prije bila na stazi 8 (dakle lijevo od trenutne pozicije). Zbog toga glava ide prvo desno. Da je npr. prethodna pozicija glave bila 50, glava bi prvo išla lijevo. Uočavamo da je razlika između lookova i scanova ta da scan ide do granica, a ne samo do krajnjih traženih staza kao look.

# Formule

(1)  natrag na: [Izlazne vrijednosti](#IzlazneVr)

(2) n (s crtom iznad) = alpha / T (s crtom iznad)

(3)

(4) n (s crtom iznad) = =

(5) T (s crtom iznad) =

(6) ili

(7) te

VAŽNO: Izrazi 3, 4,5 i 6 vrijede samo uz Poissonovu raspodjelu za broj dolazaka i eksponencijalnu raspodijelu za vrijeme trajanja obrade, dok izrazi 1,2 i 7 vrijede uvijek!

rješenje I-node zadatka -> 12 MB + 49 KB [natrag na zadatak](#zadInode)