VAŽNA NAPOMENA:

- Ova pitanja NISU materijal za pripremu kolokvijuma, već materijal za proveru znanja i vežbanje zadataka.
- Za pripremu kolokvijuma koristite PDF materijale koji su raspoloživi na stranici predmeta (sekcija: predavanja).

6.1. Navedite osnovne funkcije koje operativni sistem treba da postigne vezane za upravljanje memorijom.

Operativni sistem treba da:

- vodi evidenciju o korišćenju memorije,
- dodeljuje memoriju procesima kad je zahtevaju,
- oduzme memoriju procesima kad im je više nepotrebna,
- obavlja *swap* ukoliko operativna memorija nije dovoljnog kapaciteta za smeštaj svih procesa.

6.2. Glavni program (GP, veličine 60KB) poziva u datom redosledu sledeće segmete programa: A (20KB), B (50K) i C (40KB). Segment B u jednom trenutku poziva segment D (40KB), koji opet poziva segment E (70KB). Segment C u jednom trenutku poziva segmet F (120KB).

a. Skicirati kako se za dati primer može primeniti tehnika preklapanja (*overlay*)?

b. Koliko je najmanje radne memorije potrebno za izvršavanje ovog programa?

c. Koliko bi memorije bilo potrebno bez primene tehnike preklapanja?

(a) overlay 1: GP+A = 60K+20K = 80K

overlay 2: GP+B+D+E= 60+50+40+70 = 220K

overlay 3: GP+C+F = 60+40+120 = 220K

(b) max (80, 220, 170) = 220K

(c) GP+A+B+C+D+E+F = 60+20+50+40+40+70+120 = 400K

6.3. U čemu je razlika između logičkih i fizičkih adresa?

Adresa koju generiše procesorka instrukcija je logička adresa, a adresa same memorijske jedinice je fizička adresa. Logičke i fizičke adrese su iste u fazi prevođenja i u fazi punjenja, ali se razlikuju u fazi izvršavaja programa. Skup svih logičkih adresa koje generie program je logički (virtuelni) adresni prostor, a skup svih fizičkih adresa koje njima odgovaraju je fizički adresni prostor.

6.4. a. Gde se obavlja mapiranje virtuelnog adresnog prostora u fizički?

b. Objasnite najprostiju šemu MMU sa relokacionim registrom.

(a) Mapiranje virtuelnog adresnog prostora u fizički obavlja hardverski uređaj koji se naziva MMU (Memory-Management Unit).

(b) Relokacioni registar definiše adresu fizičkog početka programa. Svaka logička adresa koju generiše program se sabira sa vrednošću relokacionog registra, i tako se dobija fižička adresa. Logički adresni prostor koji se nalazi u opsegu [0, max] mapira se u opseg [R+0, R+max] fizičkog adresnog prostora.

6.5. a. Šta je interna fragmentacija?

b. Šta je eksterna fragmentacija?

c. Da li je eksterna fragmentacija karakteristična za kontinualnu ili diskontinualnu alokaciju memorije?

d. Kako se eksterna fragmentacija može smanjiti?

(a) Unutrašnja (interna) fragmentacija se javlja ukoliko se procesu dodeli deo memorije (na primer, stranica) veća od memorije koju proces zahteva. Preostala memorija ostaje neiskorišćena sve dok proces ne oslobodi svu memoriju koju je zauzeo.

(b) Spoljašnja (eksterna) fragmentacija je karakteristična za sisteme sa kontinualnom dodelom memorije. Proces se ne može dodeliti određena količinu memoriju, jer na sistemu ne postoji dovoljno veliki kontinualni memorijski blok (bez obzira što ukupna količina slobodne memorije premašuje zahteve procesa).

(c) Karakteristična je za kontinualnu alokaciju.

(d) Eksterna fragmentacija se može smanjiti kompakcijom memorije (povremenim premeštanjem sadržaja memorije u cilju dobijanja većih šuplljina). Kompakcija je moguća ako je relokacija programa dinamička i ako se radi u vreme izvršavanja.

6.6. Odredite kako će (a) *First fit*, (b) *Best fit* i (c) *Worst fit* algoritmi za dodelu memorije dodeliti memorijske particije od 100K, 500K, 200K, 300K i 600K (navedene redom kojim se nalaze na sistemu) procesima koji zahtevaju 212K, 417K, 112K i 426K memorije.

d. Koji algoritam obezbeđuje najefikasnije iskorišćenje memorije?

(a) *First fit*:

- 212K se smešta u particiju veličine 500K, pri čemu ostaje prazna particija veličine 500K-212K=288K. Posle dodele, sledeće particije su prazne (redom): 100K, 288K, 200K, 300K, 600K. Algoritam je obavio dva poređenja.
- 417K se smešta u particiju veličine 600K, pri čemu ostaje prazna particija veličine 600K-417K=183K. Posle dodele, sledeće particije su prazne (redom): 100K, 288K, 200K, 300K, 183K. Algoritam je obavio 5 poređenja.
- 112K se smešta u particiju veličine 288K, pri čemu ostaje prazna particija veličine 288K-112K=176K. Posle dodele, sledeće particije su prazne (redom): 100K, 176K, 200K, 300K, 600K. Algoritam je obavio dva poređenja.

- Procesu koji zahteva 426K ne može se dodeliti memorija zbog eksterne fragmentacije. Algoritam je obavio 5 poređenja.

  Proces koji zahteva 426K ne može odmah da se izvrši i mora da sačeka da neki drugi proces oslobodi memoriju. Obavljeno je ukupno 14 poređenja između veličine memorije koju proces zahteva i slobodnih particija.

(b) *Best fit*:

- 212K se smešta u particiju veličine 300K, pri čemu ostaje prazna particija veličine 300K-212K=88K. Posle dodele, sledeće particije su prazne (redom): 100K, 500K, 200K, 88K, 600K.
- 417K se smešta u particiju veličine 500K, pri čemu ostaje prazna particija veličine 500K-417K=83K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 200K, 88K, 600K. Algoritam je obavio 5 poređenja.
- 112K se smešta u particiju veličine 200K, pri čemu ostaje prazna particija veličine 200K-112K=88K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 88K, 88K, 600K. Algoritam je obavio 5 poređenja.
- 426K se smešta u particiju veličine 600K, pri čemu ostaje prazna particija veličine 600K-426K=174K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 88K, 88K, 174K. Algoritam je obavio 5 poređenja.

  Svaki proces dobija zahtevanu memoriju. Obavljeno je ukupno 20 poređenja između veličine memorije koju proces zahteva i slobodnih particija.

(c) *Worst fit*

- 212K se smešta u particiju veličine 600K, pri čemu ostaje prazna particija veličine 600K-212K=388K. Posle dodele, sledeće particije su prazne (redom): 100K, 500K, 200K, 300K, 388K. Algoritam je obavio 5 poređenja.
- 417K se smešta u particiju veličine 500K, pri čemu ostaje prazna particija veličine 500K-417K=83K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 200K, 300K, 388K. Algoritam je obavio 5 poređenja.
- 112K se smešta u particiju veličine 388K, pri čemu ostaje prazna particija veličine 388K-112K=278K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 200K, 300K, 278K. Algoritam je obavio 5 poređenja.
- Procesu koji zahteva 426K ne može se dodeliti memorija zbog eksterne fragmentacije. Algoritam je obavio 5 poređenja.

  Proces koji zahteva 426K ne može odmah da se izvrši i mora da sačeka da neki drugi proces oslobodi memoriju. Obavljeno je ukupno 20 poređenja između veličine memorije koju proces zahteva i slobodnih particija.

(d) U ovom slučaju *Best fit* algoritam obezbeđuje najbolje iskorišćenje memorije.

6.7. Data je tabela stranica jednog procesa na sitemu sistemu sa veličinom stranice 2KB.

stranica	**okvir**
0	1
1	4
2	3
3	7
4	12

Koje fizičke adrese odgovaraju sledećim logičkim adresama: (a) 251, (b) 3129, (c) 10000, (d) 6066 ?

broj stranice = ceo broj količnika logičke adrese i veličine stranice

ofset = ostatak pri deljenju logičke adrese i veličine stranice

(a) Logička adresa 251

broj stranice=[251/2048]=0 $\Rightarrow$ okvir=1, ofset=251%2048=251, fizička adresa=1x2048+251=2299

(b) Logička adresa 3129

broj stranice=[3129/2048]=1 $\Rightarrow$ okvir=4, ofset=3129%2048=1081, fizička adresa=4x2048+1081=9273

(c) Logička adresa 10000

broj stranice=[10000/2048]=4 $\Rightarrow$ okvir=12, ofset=10000%2048=1808, fizička adresa=12x2048+1808=26384

(d) Logička adresa 6066

broj stranice=[6066/2048]=2 $\Rightarrow$ okvir=3, ofset=6066%2048=1970, fizička adresa=3x2048+1970=8114

6.8. Data je sledeća tabela segmenata:

Segment	**Base**	**Length**
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Koje fizičke adrese odgovaraju sledećim logičkim adresama: (a) 0,430, (b) 1,10, (c) 2,500, (d) 3,400, (e) 4,112?

(a) 219 + 430 = 649

(b) 2300 + 10 = 2310

(c) nelegalna referenca (premašena vrednost limit registra)

(d) 1327 + 400 = 1727

(e) nelegalna referenca (premašena vrednost limit registra)

6.9. Četiri procesa izvršavaju isti editor teksta, realizovan kao *re-entrant* procedura veličine 20KB. Za privatne podatke potrebno je 3KB. Odrediti koliko je memorije potrebno i koje su osobine dodeljene memorije po pitanju kontinulanosti na sitemu sa:

a. multiprogramiranjem sa particijama promenljive veličine,

b. straničenjem (veličina stranice je 2KB),

c. segmentacijom.

(a) Za svaki proces je potrebno po 23KB kontinuilanog prostora (20KB za kod i 3KB za podatke). Ukupno je potrebno 4·23KB=92KB memorije.

(b) Potrebno je 10 stranica za kod (10·2KB=20KB) koji će procesi deliti i po dve stranice za podatke za svaki proces (4·2·2KB=4·4KB). Svaki proces koristi 3KB stranica dodeljenih za podatke, tako da ostaje 1KB neiskorišćen. Znači, ukupno je potrebno 18 stranica (36KB) koje se mogu nalaziti bilo gde u memoriji. Pri tome, imamo 4·1KB=4KB interne fragmentacije.

(c) Svaki proces zahteva dva kontinualna segmenta: deljivi kod segment veličine 20KB i segment veličine 3KB za podatke. Znači, ukupno je potrebno pet kontinualnih segmenata: jedan veličine 20KB i četiri segmenta veličine 3KB. Pri tom nema interne fragmentacije.

7.1. Koji je hardver neophodan za realizaciju tehnike učitavanja stranica na zahtev?

1. Tabela stranica sa bitom validnosti koji opisuje trenutni položaj stranica.
2. Sekundarna memorija (disk) za smeštaj stranica koje nisu prisutne u memoriji.

7.2. a. Pod kojim okolnostima dolazi do PF prekida (*page fault*)?

b. Šta radi operativni sistem kada dođe do PF prekida?

(a) Do PF prekida dolazi ukoliko proces želi da pristupi stranici koja se trenutno ne nalazi u fizičkoj memoriji.

(b) Operativni sistem proverava da li je pristup memoriji validan i prekida proces ukoliko nije. Ukoliko je pristup validan, operativni sistem traži slobodni okvir i inicira ulazno-izlaznu operaciju koja učitava stranicu sa diska u slobodni okvir. Posle toga se ažurira tabela stranica i ponovo izvršava instrukcija koja je izazvala prekid.

7.3. Dat je proces kome je dodeljeno m inicijalno praznih okvira? Proces želi da pristupi određenim stranicama koje se nalaze na disku. U nizu referenci dužine p nalazi se n različitih stranica ($m<n<p$).

a. Koliko će se najmanje PF prekida dogoditi?

b. Koliko će se najviše PF prekida dogoditi?

(a) *n*

(b) *p-m*

7.4. Pretpostavite da sistem na kom je implementirana tehnika učitavanja stranica na zahtev čuva tabelu stranica u registrima. Ukoliko ima praznih okvira ili ukoliko stranica koja se žrtvuje nije modifikovana, PF prekid se može opslužiti za 8msec. Ukoliko je stranica koja se žrtvuje modifikovana, opsluživanje PF prekida traje 20msec. Pristup lokaciji u fizičkoj memoriji traje 0.1μsec. Pretpostavite da su stranice koje se žrtvuju u 70% slučajeva modifikovane. Odredite gornju granicu verovatnoće pojave PF prekida koja će efektivno vreme pristupa održati manjim od 0.2μsec.

$t_{EA} = 0.2\mu sec = (1-P)\cdot 0.1\mu sec + (0.3p)\cdot 8msec + (0.7p)\cdot 20msec$

$p \cong 0.000006$

7.5. Data je dvodimenzionalna matrica definisana na sledeći način:

```
int A[ ][ ] = new int[2048][2048];
```

Proces koji upravlja matricom ima na raspolaganju tri okvira (veličina stranice je 4KB). Promenljiva tipa int zauzima jednu memorijsku reč veličine 2B. Pod pretpostavkom da se kod nalazi u prvom okviru, da su ostala dva okvira inicijalno prazna i da se zamena stranica obavlja po LRU algoritmu, odredite koliko PF prekida generišu sledeće petlje za inicijalizaciju matrice:

a.
```
for (int i = 0; i < 2048; i++)
    for (int j = 0; j < 2048; j++)
        A[i][j] = 0;
```

b.
```
for (int j = 0; j < 2048; j++)
    for (int i = 0; i < 2048; i++)
        A[i][j] = 0;
```

(a) 1024 PF prekida

(b) 1024 x 1024 PF prekida.

7.6. Posmatrajte sistem virtuelne memorije sa n okvira koji su u početnom trenutku prazni. Dat je sledeći niz od 20 uzastopnih memorijskih referenci u vremenu: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Skicirajte stanje u okvirima i odredite koliko će se PF prekida dogoditi ukoliko se zamena stranica obavlja po: LRU, FIFO i optimalnom algoritmu, za sledeće slučajeve: (a) n=3, (b) n=4 i (d) n=5.

(a) n=3, LRU: PF=15

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	PF	PF	PF	PF	PF	h	PF	PF	PF	h	PF	PF	h	h	PF
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6

n=3, FIFO: PF=16

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	PF	PF	PF	PF	PF	h	PF	PF	PF	h	PF	PF	h	PF	PF
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3

n=3, Optimalni algoritam: PF=11

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	PF	PF	h	h	PF	PF	h	h	PF
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	6
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1

(b) n=4, LRU: PF=10

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	PF	PF	PF	h	h	PF	H	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1

n=4, FIFO: PF=14

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	PF	PF	h	PF	PF	PF	h	PF	PF	h	PF	h
1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
	2	2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	7	3	3
		3	3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6
			4	4	4	4	4	4	1	1	1	1	1	1	2	2	2	2	2

n=4, Optimal: PF=8

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	PF	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6

(c) n=5, LRU: PF=8

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	PF	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7

n=5, FIFO: PF=10

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	PF	PF	PF	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7

n=5, Optimalni algoritam: PF=7

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	6	6	6	6	6	6	6	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7

8.1. Šta je apsolutna a šta relativna putanja datoteke? Navedite primer apsolutne i relativne putanje na Linux sistemima.

Putanja datoteke je apsolutna, ako je izražena u odnosu na početni direktorijum (npr. /home/jonhsmith), a relativna ako je izražena u odnosu na bilo koji drugi direktorijum, osim početnog (npr. backup/disk1).

8.2. a. Šta je hard a šta simbolički link na Linux sistemima ?

b. Koji od prethodno pomenutih linkova mogu ukazivati na nepostojeće objekte sistema datoteka i direktorijume ?

(a) Hard link je alternativno ime datoteke, odnosno alternativni ukazivač na i-node datoteke. Simbolički link je prečica ka objektu u sistemu datoteka, odnosno zaseban objekat sistema datoteka koji koristi jedan i-node i jedan blok podataka u kom je zapisana lokacija originalnog objekta.

(b) Simbolički.

8.3. Interpretirajte vlasničke odnose i prava pristupa za sledeću datoteku na UNIX operativnom sistemu:

```
$ ls -l file1
-rw-r--r--  1  u1  g1  509  Mar 10 17:21  file1
```

Reč o običnoj, regularnoj datoteci. Vlasnik datoteke dat1 je korisnik u1, a prava korisnika u1 u odnosu na datoteku su rw-, što znači da je može čitati i modifikovati, ali je ne može izvršavati. Datoteka dat1, koja pripada grupi g1; grupno pravo je r--, što znači da svi korisnici koji pripadaju grupi g1 mogu da pročitaju datoteku, ali je ne mogu modifikovati ni izvršavati. Pravo za ostatak sveta je r--, što žnači da svi ostali mogu čitati datoteku, ali je ne mogu modifikovati niti izvršavati.

8.4. Posmatrajte datoteku veličine 100 blokova, čiji se kontrolni blok nalazi u memoriji (u slučaju indeksne alokacije, pretpostaviti da je i indeksni čvor u memoriji računara). Odredite koliko je disk I/O operacija potrebno za izvršenje sledećih operacija na sistemima organizovanim pomoću metoda kontinualne alokacije, vezivanja blokova i indeksnih čvorova. Pretpostavite da se u slučaju kontinualne alokacije datoteka može nastaviti samo sa kraja i da se blok, koji sadrži informacije za dodavanje u datoteku, nalazi u memoriji.

a. Blok se dodaje na početak datoteke.

b. Blok se dodaje na sredinu datoteke.

c. Blok se dodaje na kraj datoteke.

d. Blok se uklanja sa početka datoteke.

e. Blok se uklanja iz sredine datoteke.

f. Blok se uklanja sa kraja datoteke.

	Kontinualna alokacija	Vezivanje blokova	Metoda indeksnih čvorova
(a)	201	1	1
(b)	101	52	1
(c)	1	3	1
(d)	198	1	0
(e)	98	52	0
(f)	0	100	0

8.5. Na FAT32 sistemu datoteka, sa klasterima veličine 2KB, nalazi se datoteka veličine 9.123 bajtova. Datoteka je redom smeštena u sledećim klasterima: 5, 10, 30, 40, 75.

a. Koliko blokova veličine 512 bajtova datoteka zauzima, a koliko blokova koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Ako je kapacitet sistema datoteka 10 GB, koliko je velika FAT tabela?

(a) Datoteka zauzima 5 klastera, a to je 5·2KB/512B= 20 blokova .

Datoteka koristi 9123/512 = 17,8 = 18 blokova, pri čemu je neiskorišćeno 93 bajta 18-tog bloka.

(b) Interna fragmentacija: 5·2048–9123= 1117 bajtova.

(c) broj klastera = kapacitet sistema datoteka / veličina klastera,

broj klastera = 10GB / 2KB = 5·1024·1024 = 5.242.880,

veličina FAT tabele = broj klastera · broj bajtova za alokaciju klastera,

veličina FAT tabele = 5242880 · 4B (32bit) = 20MB.

8.6. Na FAT16 sistemu datoteka, sa klasterima veličine 8KB, nalazi se datoteka veličine 22.345 bajtova. Datoteka je redom smeštena u sledećim klasterima: 6, 12, 15.

a. Koliko blokova veličine 512 bajtova datoteka zauzima, a koliko blokova koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Ako je kapacitet sistema datoteka 1 GB, odredite veličinu FAT tabele.

d. U kom bloku područja podataka se nalazi 5.699-ti bajt datoteke? Pretpostavite da se bajtovi datoteke, klasteri i blokovi u području podataka numerišu od 0 do N-1.

(a) Datoteka zauzima 3 klastera, a to je 3·8KB/512B= 48 blokova

Datoteka koristi 22.345/512 = 43,64 = 44 blokova, pri čemu je neiskorišćeno 475 bajtova 44-tog bloka.

(b) Interna fragmentacija: 3·8192–22.345= 2.231 bajtova.

(c) broj klastera = kapacitet sistema datoteka / veličina klastera,

broj klastera = 1·GB / 8KB = 0.125·1024·1024 = 131.072,

veličina FAT tabele = broj klastera · broj bajtova za alokaciju klastera,

veličina FAT tabele = 131.072 · 2B (16bit) = 250MB.

(d) Odredimo najpre u kom elementu klasterskog niza se nalazi traženi bajt:

5.699/512= 11,8 - to je 12 blok datoteke, odnosno 11 ti kad se broji od nule.

11/16 = 0 - nulti ulaz u *chain* kada se od nule, dakle prva pozicija, što znači *cluster* br. 6

ofset = 11%16=11 - treći blok, klaster 6 (blokovi se broje od 0)

Područje sa podacima:

cluster 0	0	1	2	3	...	10	11	12	13	14	15
cluster 1	16	17	18	19	...	26	27	28	29	30	31
...											
...											
cluster 6	96	97	98	99	...	106	107				

Blok sa podacima = broj klastera · veličina klastera + ofset

Blok sa podacima = 6 · 16 +11 = 107

8.7. Dat je UNIX sistem datoteka sa veličinom sistemskog bloka 1KB i indeksnim čvorovima u kojima se nalazi 10 direktnih pokazivača, jedan indirektni, jednan dupli indirektni i jednan trostruki indirektni pokazivač. U sistemu datoteka nalazi se datoteka veličine 7.899 bajtova, čiji je niz direktinih pokazivača u indeksnom čvoru: 2, 12, 55, 60, 76, 80, 111, 321, free, free.

a. Koliko blokova diska veličine 512B datoteka zauzima, a koliko efektivno koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Odredite najveću veličinu datoteke koja se može adresirati samo direktnim pokazivačima.

d. U kom bloku područja podataka se nalazi 3.144-ti bajt datoteke? Pretpostavite da su direktni pokazivači, sistemski blokovi, blokovi u području podataka i bajtovi datoteke numerisani od 0 do N-1.

(a) Datoteka zauzima 8 sistemskih blokova, odnosno 8·1KB = 8KB, što je ukupno 16 blokova diska.

Datoteka koristi 7.899/512 = 15,4 = 16 blokova, pri čemu je neiskorišćeno 293 bajta 16-tog bloka.

(b) Interna fragmentacija: 8·1024–7899= 239.

(c) Najveća veličina datoteke = broj direktnih pokazivača · veličina sist. bloka

Najveća veličina datoteke = 10 · 1KB = 10KB

(d) 3144/512= 6.14 = 7 - Traženi bajt se nalazi u sedmom bloku datoteke (blok broj 6 kad se broji od nule)

Najpre se određuje koji direktni pokazivač upućuje na šesti blok:

DP = blok datoteke / veličina sistemskog bloka,

sitemski blok = 2 bloka diska,

DP = 6/2 = 3 - bloku odgovara pokazivač br 3. (četvrti po redu u indeksnom čvoru). Četvrti pokazivač ukazuje na sistemski blok SB = 60,

offset = 6%2 = 0 (blok 0, odnosno prvi blok, u tom sistemskom bloku 60),

Blok (područje sa podacima) = SB · veličina SB + offset,

veličina SB = 1KB = 2 bloka na disku,

Blok (područje sa podacima) = 60 · 2 +0 = 120.

8.8. Dat je UNIX sistem datoteka sa veličinom sistemskog bloka 8KB i indeksnim čvorovima u kojima se nalazi 10 direktnih pokazivača, jedan indirektni, jednan dupli indirektni i jednan trostruki indirektni pokazivač. U sistemu datoteka nalazi se datoteka veličine 22.045 bajtova, čiji je niz direktinih pokazivača u indeksnom čvoru: 7, 22, free, free, free, free, free, free, free.

a. Koliko blokova diska veličine 512B datoteka zauzima, a koliko efektivno koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Odredite najveću veličinu datoteke koja se može adresirati samo direktnim pokazivačima.

d. U kom bloku područja podataka se nalazi 15.600-ti bajt datoteke? Pretpostavite da su direktni pokazivači, sistemski blokovi, blokovi u području podataka i bajtovi datoteke numerisani od 0 do N-1.

(a) Datoteka zauzima 3 sistemska bloka, odnosno 3·8KB = 24KB, što je ukupno 48 blokova diska.

Datoteka koristi 22.045/512 = 43,05 = 44 blokova, pri čemu je neiskorišćeno 483 bajta 44-tog bloka.

(b) Interna fragmentacija: 3·8192–22.045= 2.531 bajtova.

(c) Najveća veličina datoteke = broj direktnih pokazivača · veličina sist. bloka.

Najveća veličina datoteke = 10 · 8KB = 80KB.

(d) 15600/512= 30.1 = 31 - Traženi bajt se nalazi u 31. bloku datoteke (blok broj 30 kad se broji od nule).

Najpre se određuje koji direktni pokazivač upućuje na blok 30:

DP = blok datoteke / veličina sistemskog bloka,

sitemski blok = 2 bloka diska,

DP = 30/16 = 1 - bloku odgovara pokazivač br 1. (drugi po redu u indeksnom čvoru). Drugi pokazivač ukazuje na sistemski blok SB = 22.

offset = 30%16 = 14 (blok 14 u sistemskom bloku 22),

Blok (područje sa podacima) = SB · veličina SB + offset,

veličina SB = 8KB = 16 bloka na disku,

Blok (područje sa podacima) = 22 · 16 +14 = 366.