Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1 (SI2OS1, IR2OS1)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Treći, maj 2012.
Datum: 15.6.2012.

Treći kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 1,5 sat. Dozvoljeno je korišćenje literature.

Zadatak 1 _____/10 *Zadatak 3* _____/10
Zadatak 2 _____/10

Ukupno: _____/30 = _____% = _____/10

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

1. (10 poena) Ulaz/izlaz

Neki sistem organizuje keš blokova sa diska na sledeći način. Keš čuva najviše `CACHESIZE` blokova veličine `BLKSIZE` u nizu `diskCache`. Svaki ulaz *i* u nizu `diskCacheMap` sadrži broj bloka na disku koji se nalazi u ulazu *i* keša. Vrednost 0 u nekom ulazu niza `diskCacheMap` označava da je taj ulaz prazan (u njega nije učitan blok). Data je sledeća implementacija keša:

```
typedef ... Byte; // Unit of memory
typedef ... BlkNo; // Disk block number
const int BLKSIZE = ...; // Disk block size in Bytes
const int CACHESIZE = ...; // Disk cache size in blocks

BYTE diskCache [CACHESIZE][BLKSIZE]; // Disk cache
BlkNo diskCacheMap [CACHESIZE]; // The contents of disk cache. 0 for empty

int diskCacheCursor = 0; // FIFO cursor for eviction/loading

Byte* getDiskBlock (BlkNo blk) {
  // Search for the requested block in the cache and return it if found:
  for (int i=0; i<CACHESIZE; i++) {
    if (diskCacheMap[i]==blk) return diskCache[i];
    if (diskCacheMap[i]==0) break;
  }
  // Not found.
  // If there is a block to evict, write it to the disk:
  if (diskCacheMap[diskCacheCursor]!=0)
    diskWrite(diskCacheMap[diskCacheCursor],diskCache[diskCacheCursor]);
  // Load the requested block:
  diskCacheMap[diskCacheCursor] = blk;
  diskRead(blk,diskCache[diskCacheCursor]);
  Byte* ret = diskCache[diskCacheCursor];
  diskCacheCursor = (diskCacheCursor+1)%CACHESIZE;
  return ret;
}
```

Operacije `diskRead` i `diskWrite` vrše sinhrono čitanje, odnosno upis datog bloka sa diska:

```
void diskRead(BlkNo block, Byte* toBuffer);
void diskWrite(BlkNo block, Byte* fromBuffer);
```

Ukoliko je keš pun, a treba učitati novi blok, iz keša se izbacuje blok iz ulaza na koji ukazuje kurzor `diskCacheCursor` koji se pomera u krug, tako da je izbacivanje (engl. *eviction*) po FIFO (*first-in-first-out*) principu.

Funkcija `getDiskBlock` vraća pokazivač na deo memorije u kome se nalazi učitan traženi blok diska, pri čemu se taj blok učitava u keš (uz prethodno snimanje eventualno izbačenog bloka) ukoliko taj blok već nije u kešu. Ovu funkciju koristi ostatak sistema za pristup blokovima diska.

Ova implementacija keša je nedovoljno efikasna jer se neki blok sa diska može smestiti u bilo koji ulaz keša i zato pronalaženje bloka u punom kešu često zahteva obilaženje velikog dela niza `diskCacheMap`. Zbog toga treba promeniti implementaciju ovog keša tako da se `diskCacheMap` organizuje kao heš (*hash*) tabela. Disk blok broj *b* se smešta u ulaz *hash*(*b*)=*b* mod CACHESIZE, ukoliko je taj ulaz slobodan. U slučaju da nije (tj. u slučaju kolizije), taj blok se smešta u prvi naredni slobodan blok (u krug, po modulu `CACHESIZE`; rešavanje kolizije otvorenim adresiranjem).

Dati ovako izmenjenu funkciju `getDiskBlock`.

2. (10 poena) Interfejs fajl sistema

U nekom fajl sistemu u sistemskom pozivu za otvaranje fajla proces navodi da li će fajl samo čitati ili ga i na bilo koji način menjati. U zavisnosti od toga, taj sistemski poziv zaključava fajl sa jednim od dve vrste ključa. Ako se fajl otvara samo za čitanje, fajl se zaključava deljenim ključem; ako se fajl otvara za izmenu, zaključava se ekskluzivnim ključem. Ukoliko poziv ne može da se izvrši zbog toga što ključ ne može da se dobije, poziv se otkazuje bez izmena u fajl sistemu i vraća se greška.

Procesi A, B, C i D izvršavaju sistemske pozive otvaranja i zatvaranja istog fajla u sledećem redosledu (neki proces izvršava poziv zatvaranja fajla samo ako ga je uspešno otvorio):

1) A: open(READ)
2) B: open(WRITE)
3) C: open(READ)
4) A: close
5) C: close
6) D: open(WRITE)

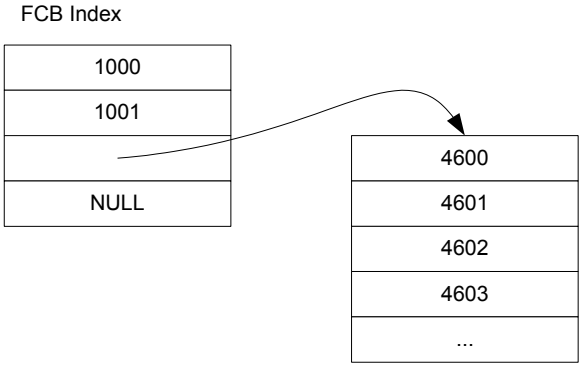
Koje od ovih operacija će se izvršiti uspešno, a koje neuspešno?
Precizno obrazložiti odgovor!

Odgovor:

3. (10 poena) Implementacija fajl sistema

Neki fajl sistem primenjuje kombinovanu tehniku indeksirane alokacije sadržaja fajla. U FCB fajla nalaze se dva ulaza koji predstavljaju indeks nultog nivoa (direktni pokazivači na dva prva bloka sadržaja fajla) i još dva ulaza koji ukazuju na blokove sa indeksima prvog nivoa.

Na slici je prikazan deo FCB nekog fajla. Blok je veličine 1KB, a broj bloka (svaki ulaz u indeksnom bloku) zauzima 4 bajta. Bajtovi sadržaja fajla se broje počev od 0.



U kom bloku na disku se nalazi bajt sadržaja ovog fajla sa datim rednim brojem (pored konačnog odgovora, dati i celu računicu):

a)(3) 2000 (decimalno)?

Odgovor: _____

Postupak i obrazloženje:

b)(3) 3570 (decimalno)?

Odgovor: _____

Postupak i obrazloženje:

c)(4) Kolika je maksimalna veličina fajla koju dozvoljava ovaj sistem?

Odgovor: _____

Postupak i obrazloženje: