Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Treći, jun 2024.
Datum: 9. 6. 2024.

Treći kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. 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

Korišćenjem biblitečnih funkcija *popen*, *dup2* i *fileno* implementirati funkciju *redirect* čiji je potpis dat i koja pokreće proces dete nad programom u fajlu sa putanjom zadatom argumentom `exe`, a onda preusmerava standardni izlaz pozivajućeg (svog) procesa na standardni ulaz tog procesa deteta; u slučaju neuspeha vraća -1, u slučaju uspeha vraća 0. Biblitečna funkcija *fileno* vraća celobrojni deskriptor fajla (*fd*) koji odgovara zadatom znakovnom toku `stream`.

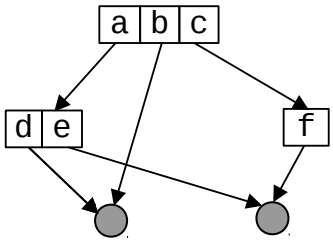
```
int fileno (FILE* stream);
int redirect (const char* exe);
```

Rešenje:

2. (10 poena) Fajl sistem

Ispisati sekvencu Unix komandi kojima korisnik, kroz CLI, može da napravi strukturu (pod)direktorijuma (na slici predstavljeni pravougaonicima) i fajlova (na slici predstavljeni kružićima) koja je prikazana na datoj slici, unutar tekućeg direktorijuma koji je inicijalno prazan; „koreni“ direktorijum prikazan na slici predstavlja taj tekući direktorijum (u njemu treba formirati ulaze *a*, *b*, i *c*). Fajlove praviti tako da im se sadržaj učitava sa konzole (unos znaka Ctrl-D označava kraj ulaznog toka na konzoli); ne pisati unos koji se unosi na konzoli za sadržaj fajla. Sve strelice na slici predstavljaju tvrde veze (*hard links*). Na raspolaganju su sledeće komande:

- `mkdir` *directory*: pravi navedeni direktorijum sa podrazumevanim pravima pristupa; pretpostaviti da su ta podrazumevana prava pristupa dovoljna da tekući korisnik uradi sve potrebne komande;
- `cat`: ukoliko nije naveden ulazni fajl, ovaj sistemski program sadržaj koji daje na svoj standardni izlaz uzima sa svog standardnog ulaza;
- `ln` *original_filename link_name*: pravi tvrdu vezu *link_name* ka fajlu *original_filename*;
- `cd` *target_dir*: menja tekući direktorijum na zadati.



Rešenje:

3. (10 poena) Fajl sistem

U implementaciji nekog fajl sistema keš struktura FCB svih otvorenih fajlova i korišćenih direktorijuma („globalna tabela otvorenih fajlova“) organizuje se u memoriji kao objekat klase `FCBCache`. U tom kešu jedan FCB čuva se u jednom ulazu predstavljenom strukturom `FCBEntry` čiji je deo definicije dat dole. Polje `id` čuva identifikator FCB-a koji je učitan u ovaj ulaz (npr. broj bloka na disku), polje `refCnt` je brojač referenci, a polje `fcb` sam sadržaj FCB-a.

```
struct FCBEntry {
  FCBID id;  // ID of the FCB stored in this entry
  unsigned long refCnt = 0; // Reference counter
  FCB fcb;  // FCB of the node
  ...
};
```

Kada neki deo kernela želi da koristi neki FCB, recimo kada proces otvara fajl, poziva se operacija `FCBCache::request(FCBID)` koja pronalazi ulaz u kome se već nalazi traženi FCB ili ga učitava ukoliko on nije u kešu (tabeli). Tom prilikom se `refCnt` tog ulaza inkrementira ili postavlja na 1 ukoliko je dati FCB tek učitan. Slično, kada taj korisnik više ne koristi taj FCB, poziva operaciju `FCBCache::release(FCBID)` koja „oslobađa“ taj FCB, ali on ostaje u kešu ukoliko ponovo bude potreban, dok se eventualno odatle ne izbaci. Keš funkcioniše tako što nekorišćeni ulazi (oni kod kojih je `refCnt==0`) ostaju u kšeu osim ukoliko se ne zahteva prostor za učitavanje novog FCB-a, a u kešu više nema slobodnih ulaza. Tada se kao „žrtva za izbacivanje“ bira onaj nekorišćeni ulaz koji je najdavnije bio korišćen – LRU (*least recently used*) politikom zamene.

Implementirati sledeće pomoćne nestatičke funkcije članice klase `FCBCache` koje se pozivaju iz implementacije navedenih operacija `FCBCache::request` i `FCBCache::relase`:

- `void FCBCache::updateLRUonRequest (FCBEntry* f)`: ova operacija poziva se iz operacije `FCBCache::request` koja je pronašla odgovarajući ulaz `f`; ona treba da po potrebi ažurira evidenciju potrebnu za LRU algoritam zamene; pre poziva ove operacije `f->refCnt` je već ažuriran (inkrementiran);
- `void FCBCache::updateLRUonRelease (FCBEntry* f)`: ova operacija poziva se iz operacije `FCBCache::release` koja oslobađa odgovarajući ulaz `f`; ona treba da po potrebi ažurira evidenciju potrebnu za LRU algoritam zamene; pre poziva ove operacije `f->refCnt` je već ažuriran (dekrementiran);
- `FCBEntry* FCBCache::getLRUVictim ()`: ova operacija poziva se iz operacije `FCBCache::request` po potrebi, ukoliko nema slobodnog mesta za učitavanje traženog FCB-a; ona treba da vrati ulaz koji je po LRU algoritmu odabran kao žrtva za zamenu i po potrebi ažurira evidenciju potrebnu za LRU algoritam zamene; ukoliko takvog nema, treba da vrati *null*.

Navesti proširenja klase `FCBCache` i strukture `FCBEntry` potrebnim članovima.

Rešenje: