Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (13E113OS2, 13S113OS2)
Nastavnik: prof. dr Dragan Milićev
Školska godina: 2016/2017. (Zadatak važi počev od januarskog roka 2017.)

Projekat za domaći rad
- Projektni zadatak –

Verzija dokumenta: 1.0

Važne napomene: Pre čitanja ovog teksta, **obavezno** pročitati opšta pravila predmeta i pravila vezana za izradu domaćih zadataka! Pročitati potom ovaj tekst **u celini i pažljivo**, pre započinjanja realizacije ili traženja pomoći. Ukoliko u zadatku nešto nije dovoljno precizno definisano ili su postavljeni kontradiktorni zahtevi, student treba da uvede razumne pretpostavke, da ih temeljno obrazloži i da nastavi da izgrađuje preostali deo svog rešenja na temeljima uvedenih pretpostavki. Zahtevi su namerno nedovoljno detaljni, jer se od studenata očekuje kreativnost i profesionalni pristup u rešavanju praktičnih problema!

Uvod

Cilj ovog zadatka jeste implementacija uprošćenog školskog sistema za alokaciju memorije. Sistem za alokaciju memorije treba da obezbedi alokaciju i dealokaciju memorije na nivou jezgra operativnog sistema. Na nivou jezgra treba obezbediti da se alokacija i dealokacija podataka samog jezgra radi brzo i kompaktno, korišćenjem sistema ploča (engl. *slab*) i sistema parnjaka (engl. *buddy*).

Opšti zahtevi

Odnos projekta i korisničke aplikacije

Tražene podsisteme treba realizovati na jeziku C/C++. Korisničku aplikaciju, koja sadrži test primere, treba povezati sa prevedenim kodom projekta u jedinstven konzolni program (.exe). U datoj aplikaciji biće prisutna i funkcija *main*.

Odnos projekta i operativnog sistema domaćina

Projekat se realizuje pod operativnim sistemom Windows 7 ili novijim, koji je u ovom slučaju operativni sistem domaćin. Izradom projekta se ni na koji način ne sme ugroziti ispravno funkcionisanje operativnog sistema domaćina. Svaki eventualni problem koji se pojavi po pokretanju projekta biće smatran kao greška pri izradi projekta. Po završetku rada, okruženje je neophodno ostaviti u neizmenjenom stanju u odnosu na trenutak pre pokretanja projekta, osim onih delova koji se namerno menjaju samim test primerom koji treba da proveri ispravnost projekta. Svi resursi u sistemu koji će biti korišćeni pri izradi projekta moraju biti korišćeni kroz odgovarajući API operativnog sistema domaćina koji je za to namenjen. Usluge operativnog sistema i samog programskog jezika vezane za alokaciju memorije nije dozvoljeno koristiti. Deo koda koji je obezbeđen u okviru postavke projekta je pažljivo napisan, i ukoliko se koristi u skladu sa uputstvom za rad, ne može prouzrokovati nikakve probleme i greške pri izvršavanju.

Zadatak: Alokacija memorije u jezgru operativnog sistema

Uvod

Za alokaciju podataka jezgro operativnog sistema koristi alokator pod nazivom sistem ploča (engl. *slab allocator*) koji će biti implementiran u okviru ovog projekta. Na početku rada alokatoru će biti dodeljen kontinualan memorijski prostor koji će alokator kontrolisati. Alokator ima sledeće ciljeve:

- alokacija malih memorijskih bafera sa ciljem da se smanji interna fragmentacija;
- keširanje često korišćenih objekata sa ciljem da se izbegne alokacija, inicijalizacija i uništavanje objekata;
- bolje iskorišćenje hardverske keš memorije poravnanjem objekata na L1 linije keš memorije.

Prvi cilj treba da se postigne održavanjem jednog skupa keševa za alociranje malih memorijskih bafera čija je moguća veličina između 2^5 i 2^{17} bajtova. Ovi keševi se nazivaju *size-N*, gde je *N* veličina potrebnog prostora za jedan bafer. Veličina malog memorijskog bafera (u bajtovima) može biti jednaka samo stepenu dvojke.

Drugi cilj se postiže održavanjem keševa često korišćenih objekata. Kada se nova ploča kreira, određeni broj objekata jezgra se alocira u njemu. Objekti se inicijalizuju ako postoji konstruktor za njih. Kada se objekat oslobodi, ostavlja se u inicijalizovanom stanju da može da bude ponovo upotrebljen.

Treći cilj se postiže poravnanjem objekata u pločama (ako ima mesta) na linije L1 hardverske keš memorije, tako da se objekti iz različitih ploča keširaju u različitim linijama hardverske keš memorije. To se postiže postavljanjem objekata na različite pomeraje u okviru ploča.

Alokator treba da obezbedi interfejs za kreiranje, uništavanje i smanjenje keševa, kao i interfejs za alokaciju/dealokaciju objekata i malih memorijskih bafera. Za potrebe testiranja alokator treba da obezbedi i interfejs za štampanje informacija o keševima. Alokator treba automatski da proširuje veličinu keša kada je to potrebno.

Alokator se inicijalizuje prilikom startovanja sistema pozivom odgovarajuće funkcije kojoj se prosleđuje pokazivač na memoriju koju alokator treba da kontroliše, kao i veličina tog prostora izraženog u broju blokova. Ploče treba da budu veličine 2^n kontinualnih blokova. Za vođenje evidencije o slobodnim i zauzetim blokovima koristiti sistem parnjaka (engl. *buddy allocator*). Alokatori smeju da koriste samo dodeljenu memoriju za čuvanje svojih podataka, drugim rečima ne sme da dinamčki alociraju memoriju korišćenjem usluga samog jezika i operativnog sistema domaćina (*malloc*, *new* itd.).

Sistem ploča

Opis keša

Za potrebe alociranja se koriste keševi za objekte svih vrsta i male memorijske bafere. Keševi treba da budu ulančani u listu. Jedan keš sadrži informacije o pločama za jednu vrstu objekata ili za male memorijske bafere. Keš treba da čuva tri liste ploča, jednu za slobodne ploče, jednu za pune ploče i jednu za sve ploče koje imaju mesta a nisu potpuno prazne. Proširenje keša se radi automatski kad su sve ploče popunjene. Smanjivanje jednog keša se radi tako što

se prazne ploče uklone, pod uslovom da od prethodnog smanjivanja nije bilo potrebe da se broj ploča poveća. Alokacija jednog malog memorijskog bafera implicitno kreira keš za tu vrstu malih memorijskih bafera.

Poravnanje objekata u pločama se radi na nivou jednog keša. Ideja je da se objekti iz različitih ploča keširaju u različitim linijama keša. Za tu svrhu data je veličina jedne hardverske keš linije izražena u bajtovima. Prilikom kreiranja keša treba proračunati koliko prostora u ploči ostaje neiskorišćeno. Broj mogućih poravnanja se dobija kao količnik veličine neiskorišćenog prostora i veličine jedne hardverske keš linije. Prvi objekt u ploči je za jednu keš liniju pomeren u odnosu na prvi objekat u prethodnoj ploči. Kada pomeraj pređe veličinu neiskorišćenog prostora pomeraj se resetuje na nulu. (Na primer: veličina jedne linije hardverskog keša je 64B a veličina slobodnog prostora je 200B. Tada je pomeraj u prvoj ploči 0, u drugoj 64, u trećoj 128, u četvrtoj 0, itd.)

Informacije koje se štampaju za jedan keš su ime, veličina jednog podatka izražena u bajtovima, veličina celog keša izraženog u broju blokova, broj ploča, broj objekata u jednoj ploči i procentualna popunjenost keša.

Operacije koje za keš treba da se obezbede su:

- inicijalizacija alokatora; prosleđuje se adresa pocetka memorijsko prostora za koji je alokator zadužen i veličina tog memorijskog prostora izražena u broju blokova;
- kreiranje jednog keša; pri kreiranju se zadaje naziv keša, veličina objekta, i opcioni konstruktor i destruktor objekata; povratna vrednost je pokazivač na ručku keša; NULL vrednost može da se prosledi umesto konstruktora i destruktora;
- smanjivanje jednog keša; povratna vrednost je broj blokova koji je oslobođen;
- brisanje jednog keša;
- alokaciju jednog objekta;
- dealokaciju jednog objekta;
- alokacija jednog malog memorijskog bafera;
- dealokacija jednog memorijskog bafera;
- štampanje informacije o kešu;
- štampanje greške prilikom rada sa kešom; povratna vrednost funkcije je 0 ukoliko greške nije bilo, u suprotnom vrednost različita od 0.

Sve date operacije su *thread-safe*, što znači da se potpuno bezbedno mogu pozivati iz konkurentnih niti. Sinhronizaciju smanjiti na minimum. Implementacija ručke za pristup kešu se ostavlja u nadležnost samog rešenja.

Keš na jeziku C

Interfejs za pristup keševima, zajedno sa potrebnim tipovima i podacima, dat je u fajlu `slab.h`.

```
// File: slab.h
#include <stdlib.h>

typedef struct kmem_cache_s kmem_cache_t;
#define BLOCK_SIZE (4096)
#defince CACHE_L1_LINE_SIZE (64)

void kmem_init(void *space, int block_num);
```

```
kmem_cache_t *kmem_cache_create(const char *name, size_t size,
                                    void (*ctor)(void *),
                                    void (*dtor)(void *); // Allocate cache
int kmem_cache_shrink(kmem_cache_t *cachep); // Shrink cache
void *kmem_cache_alloc(kmem_cache_t *cachep); // Allocate one object from cache
void kmem_cache_free(kmem_cache_t *cachep, void *objp); // Deallocate one object from cache
void *kmalloc(size_t size); // Alloacate one small memory buffer
void kfree(const void *objp); // Deallocate one small memory buffer
void kmem_cache_destroy(kmem_cache_t *cachep); // Deallocate cache
void kmem_cache_info(kmem_cache_t *cachep); // Print cache info
int kmem_cache_error(kmem_cache_t *cachep); // Print error message
```

Sistem parnjaka

Opis zadatka

Sistem parnjaka se koristi samo u priloženom rešenju. Drugim, rečima nije potrebno obezbediti interfejs za korišćenje sistema parnjaka van samog rešenja. Implementacija sistema parnjaka se ostavlja u potpunosti u nadležnost samog rešenja.

Testovi

Javni testovi

Javni test-program služi da pomogne studentima da istestiraju svoj projekat. Ovi testovi neće obavezno pokriti sve funkcionalnosti koje projekat treba da ima, ali će istestirati većinu tih funkcionalnosti. Da bi se projekat uopšte odbranio, neophodno je da projekat sa javnim testom radi u potpunosti ispravno. Studentima se preporučuje da pored javnog testa naprave i svoje testove koji će im pomoći da što bolje istestiraju svoj projekat.

Tajni testovi

Tajni testovi detaljnije testiraju sve zahtevane funkcionalnosti u različitim regularnim i neregularnim situacijama (greške u pozivu ili radu alokatora), i nisu unapred dostupni studentima.

Testovi performansi

Testovi performansi mere vreme izvršavanja pojedinačnih operacija. Ovi testovi nisu obavezni, i mogu, ali ne moraju, doneti dodatne bodove u prvom ispitnom roku posle nastave za do 10 najboljih radova odbranjenih pre roka.

Zaključak

Potrebno je realizovati opisane podsisteme prema datim zahtevima na jeziku C++. Kao integrisano okruženje za razvoj programa (engl. integrated development environment, IDE) zahteva se Microsoft Visual Studio 2015 radi kompatibilnosti sa. Testiranje se vrši u laboratorijama katedre na računarima pod operativnim sistemom Windows 10 x64.

Pravila za predaju projekta

Projekat se predaje isključivo kao jedna zip arhiva. Sadržaj arhive podeliti u dva foldera: src i h. U prvom folderu (src) treba da budu smešteni svi .cpp fajlovi koji su rezultat izrade projekta, a u drugom folderu (h) treba da budu svi .h fajlovi koji su rezultat izrade projekta. Opisani sadržaj ujedno treba da bude i jedini sadržaj arhive (arhiva ne sme sadržati ni izvršne fajlove, ni biblioteke, ni .cpp i .h fajlove koji predstavljaju bilo kakve testove, niti bilo šta što iznad nije opisano). Projekat je moguće upload-ovati više puta, ali do trenutka koji će preko e-mail liste biti objavljen za svaki ispitni rok i koji će uvek biti pre ispita. Na serveru uvek ostaje samo poslednja predata verzija i ona će se koristiti na odbrani. Za izlazak na ispit neophodno je predati projekat (prijava ispita i položeni kolokvijumi su takođe preduslovi za izlazak na ispit). Nakon isteka roka za predaju projektni zadaci se brišu sa servera, pa u slučaju ponovnog izlaska na ispit potrebno je ponovo postaviti ažurnu verziju projektnog zadataka.

Sajt za predaju projekta je http://rti.etf.rs/rti/os/os2/index.php

Zapisnik revizija

Ovaj zapisnik sadrži spisak izmena i dopuna ovog dokumenta po verzijama.

Verzija 1.0

Strana	Izmena