Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (13E113OS2, 13S113OS2)
Nastavnik: prof. dr Dragan Milićev
Asistent: Živojin Šuštran
Školska godina: 2025/2026. (Zadatak važi počev od prvog roka u 2026. god.)

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*).

Zadatak se sastoji od dva dela. Prvi deo je obavezan i za uspešnu odbranu projektnog zadatka student mora da ga uradi. Svi algoritmi i optimizacije se ostavljaju u nadležnosti samog rešenja i biće razmatrani samo u okviru testiranja performansi sistema.

Opšti zahtevi

Odnos projekta i korisničke aplikacije

Tražene podsisteme treba realizovati na jeziku C. Korisničku aplikaciju, koja sadrži test primere, prevesti nezavisno u konzolni program. U datoj aplikaciji biće prisutna i funkcija *main*. Korisnička aplikacija će napraviti određeni broj procesa korišćenjem sistemskog poziva *fork*. Korisnička aplikacija pristupa uslugama operativnog sistema jedino putem sistemskih poziva.

Odnos projekta i ostatka operativnog sistema

Dat je operativni sistem xv6[1] (prilagođen programski kod se nalazi na sajtu predmeta). Zadatak studenta je da izmeni deo operativnog sistem xv6 tako da podrži alokator koji je opisan u ovom projektu. Izradom projekta se ni na koji način ne sme ugroziti ispravno funkcionisanje ostalih delova operativnog sistema. Svaki eventualni problem koji se pojavi po pokretanju projekta biće smatran kao greška pri izradi projekta. 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. Dati kod dozvoljeno je menjati u onoj meri koja je potrebna za realizaciju traženih zahteva, ali nikako uklanjanjem već postojećih funkcionalnosti i menjanjem postojećih sistemskih poziva.

Razvojno okruženje

xv6 operativni sistem se izvršava na emulatoru u okviru operativnog sistema domaćina (Linux x64). Virtuelna mašina sa instaliranim operativnim sistemom domaćinom, emulatorom i svim potrebnim alatima za prevođenje je data na sajtu predmeta. Ista ta virtuelna mašina će biti korišćena i na odbrani projekta. Predviđeno razvojno okruženje je CLion (studenti mogu nabaviti akademsku licencu na sajtu proizvođača sa studentskim email nalogom). Uputstvo za podešavanje projekta i testiranja je dato na sajtu predmeta. S obzirom da se razvija kernel operativnog sistema standardna biblioteka C jezika nije dostupna. Dovoljno je koristi sistem sa jednim procesorom.

[1] Detaljan opis xv6 operativnog sistema se može naći u knjizi na adresi (pristupljeno: 15.12.2025. godine) https://pdos.csail.mit.edu/6.1810/2024/xv6/book-riscv-rev4.pdf

Prvi deo (20 poena)

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;

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.

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.

Prilikom startovanja sistema alokatoru se dodeljuje prostor koji treba da kontroliše. 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.

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 malog 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 se pozivaju isključivo u sistemskom modu rada procesora.

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


typedef struct kmem_cache_s kmem_cache_t;
#define BLOCK_SIZE (4096)


typedef unsigned long size_t;


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.

Drugi deo (10 poena)

Uvod

Alokator je potrebno koristiti u već postojećim podsistemima operativnog sistema. Za svaki objekat koji pravi kernel potrebno je koristiti keševe objekata. Održavanje keševa objekata (kreiranje, smanjivanje, uništavanje) je potrebno raditi na odgovarajućim mestima u kernelu i to potpada pod nadležnost samog rešenja. Za nizove koje alocira kernel (npr. bafer za prihvatanje podataka sa tastature) potrebno je koristiti alokaciju malih memorijskih bafera. U kernelu nije dozvoljeno da se ništa osim skalarnih podataka (npr. pokazivač, integer, itd.) alocira statički, bez korišćenja implementiranog alokatora.

Testovi

Javni testovi

Testiranje može da se radi na dva načina. Prvi je eksplicitno testiranje i on je potreban za prvi deo zadatka. Potrebno obezbedi sistemske pozive za API alokatora. Korisnički proces tom prilikom može da poziva funkcije alokatora (podrazumeva se i init funkciju). Tom prilikom korisnička aplikacija će direktno pozivati funkcije koje se implementiraju. Drugi način je implicitno i on je potreban za drugi deo zadatka. Korisnička aplikacija neće pozivati sistemske pozive služe za pristup alokatoru, već samo ostale sistemske pozive. Na proizvoljan način omogućiti oba načina testiranja. Za prvi način je dostupna aplikacija kao javni test na sajtu, dok za drugi način su sve aplikacije dostupne u operativnom sistemu.

Tajni testovi

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

Testovi performansi

Testovi performansi mere vreme izvršavanja procesa i efikasnost u korišćenju resursa. Ovi testovi nisu obavezni, i mogu, ali ne moraju, doneti dodatne bodove u predroku posle nastave za do 20 najboljih odbranjenih radova. Ceni se i dodatne funkcionalnosti, optimizacije i slično koje nisu tražene datom postavkom.

Zaključak

Potrebno je realizovati opisane podsisteme prema datim zahtevima na jeziku C. Testiranje se vrši u laboratorijama katedre na računarima pod operativnim sistemom Windows 11 x64. Virtuelna mašina sa sajta predmeta biće dostupna za vreme odbrane.

Pravila za predaju projekta

Projekat se predaje isključivo kao jedna zip arhiva. U arhivu smestiti samo fajlove sa kodom (.c, .cpp, .h i slične) 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 bilo kakve testove, niti bilo šta što iznad nije opisano, pogotovu ne sme sadržati repozitorijum koda. Projekat je moguće predati više puta, ali do trenutka koji će preko imejl liste biti objavljen za svaki ispitni rok i koji će uvek biti pre ispita, po pravilu prvi radni dan 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 je u slučaju ponovnog izlaska na ispit potrebno ponovo postaviti ažurnu verziju projektnog zadataka.

Sajt za predaju projekta je https://rti.etf.bg.ac.rs/domaci/index.php?servis=os2_projekat

Zapisnik revizija

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

Verzija 1.0

Strana	Izmena