Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika
Kolokvijum: Prvi, jun 2022.
Datum: 12. 6. 2022.

Prvi 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 = _____% = _____/15

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)

Posmatraju se četiri odvojena i nezavisna slučaja sadržaja tri fajla sa C kodom, `a.h`, `a.c` i `b.c`, data u tabeli i označena rednim brojem 1 do 4. U svakom od tih slučajeva se sprovodi sledeći postupak:

- najpre se prevodi fajl `a.c`; ako prevodilac prijavi grešku, postupak se prekida;
- ako u prevođenju fajla `a.c` nije bilo grešaka, prevodi se fajl `b.c`; ako prevodilac prijavi grešku, postupak se prekida;
- ako ni u prevođenju fajla `b.c` nije bilo grešaka, povezuju se fajlovi `a.o` i `b.o`.

Za svaki od datih slučajeva precizno objasniti grešku koju će prijaviti prevodilac, odnosno linker, ukoliko grešaka ima: navesti ko će prijaviti grešku i kakvog tipa je ta greška (i zbog kog simbola). Ukoliko grešaka nema, to naglasiti („nema grešaka“).

#	**a.h**	**a.c**	**b.c**
1	`int f ();`	`#include "a.h"` `int f ();`	`#include "a.h"` `int main () { return f(); }`
2	`extern int x;`	`#include "a.h"` `int x = 0;`	`#include "a.h"` `int main () { return x; }`
3	`int x = 0;`	`#include "a.h"` `int x = 0;`	`extern int x;` `int main () { return x; }`
4	`int x = 0;`	`#include "a.h"` `extern int x;`	`#include "a.h"` `int main () { return x; }`

Odgovori:

#1: _____

#2: _____

#3: _____

#4: _____

2. (10 poena)

U nekom sistemu jedan objekat klase `FreeFrames`, čiji je interfejs dat dole, vodi evidenciju o slobodnim okvirima memorije predviđene za smeštanje stranica procesa. Objekat ove klase inicijalizuje se zadavanjem kontinualnog memorijskog prostora od `size` susednih stranica počev od adrese `mem`. Veličina stranice je `PAGE_SIZE` (u jedinicama `sizeof(char)`). Operacija `alloc` vraća jedan slobodan okvir i izbacuje ga iz evidencije slobodnih okvira. Operacija `free` dodaje slobodni okvir na koga ukazuje argument u evidenciju slobodnih okvira.

Implementirati klasu `FreeFrames` u potpunosti tako da operacije `alloc` i `free` budu najjefikasnije moguće ($O(1)$).

```
class FreeFrames {
public:
  FreeFrames (void* mem, size_t size);
  void* alloc ();
  void  free (void* frame);
};
```

Rešenje:

3. (10 poena)

Neki sistem ima segmentno-straničnu organizaciju memorije. Virtuelna adresa je 32-bitna, adresibilna jedinica je bajt, a maksimalna veličina fizičkog segmenta je 16 MB. Stranica je veličine 64 KB. Fizički adresni prostor je veličine 4 GB. Jedan ulaz u SMT-u zauzima dve 32-bitne reči; u prvoj (nižoj) su prava pristupa *rwx* u najviša tri bita, a granica fizičkog segmenta (*limit* u opsegu od 0 do maksimalno dozvoljenog broja stranice koji se sme adresirati) u najnižim bitima; druga (viša) reč ulaza u SMT-u sadrži 32-bitni pokazivač na PMT tog segmenta.

Operativni sistem alocira stranice na zahtev, tako da pri kreiranju procesa ne alocira nijednu stranicu. Vrednost 0 u polju za pokazivač na PMT u deskriptoru segmenta u SMT-u, kao i u deskriptoru stranice u PMT-u označava da preslikavanje nije moguće.

Jedan alociran logički segment procesa opisan je deskriptorom tipa `SegDesc` u kom su, između ostalog, sledeća polja:

- `unsigned startAddr`: početna virtuelna adresa logičkog segmenta, svakako poravnata na početak fizičkog segmenta;
- `unsigned size`: veličina logičkog segmenta u bajtovima (može biti i veća od maksimalne veličine fizičkog segmenta);
- `unsigned short rwx`: prava pristupa za ceo logički segment u tri najniža bita.

Implementirati sledeću funkciju:

```
int initSegment (SegDesc* sd, unsigned long smt[][2]);
```

Ovu funkciju poziva kod kernela kada inicijalizuje SMT novokreiranog procesa za svaki logički segment sa datim deskriptorom `sd`. Na već alociran SMT ukazuje `smt`. Potrebno je inicijalizovati ulaze u SMT i napraviti pridružene PMT za sve korišćene fizičke segmente. Veličine tipova su sledeće: `int` – 32 bita, `long` – 64 bita, `short` – 16 bita.

Statička funkcija `PMT::alloc()` alocira prostor u memoriji kernela za smeštanje PMT jednog segmenta, inicijalizuje ceo taj PMT na 0 i vraća 32-pokazivač taj PMT. Kako bi se obezbedilo da bude dovoljno prostora za sve PMT-ove potrebne za sve fizičke segmente koje zauzima jedan logički segment, funkcija `initSegment` treba najpre da proveri da li ima dovoljno prostora za njih. Ovu proveru obavlja statička funkcija `PMT::reserve(int segs)` koja proverava da li ima dovoljno prostora za PMT-ove `segs` segmenata, rezerviše taj prostor (da bi narednih `segs` poziva `PMT::alloc` sigurno uspelo) i vraća 0 u slučaju uspeha, a negativnu vrednost u slučaju nedostatka prostora. Funkcija `initSegment` treba da vrati 0 u slučaju uspeha, a negativnu vrednost u suprotnom.

R
e
š
e
n
j