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: Drugi, maj 2022.
Datum: 8. 5. 2022.

Drugi 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)

Koristeći samo sistemske pozive *fork*, *wait* i *exit*, kao i funkciju *printf*, napisati C program koji pronalazi maksimalnu vrednost svih elemenata ogromne celobrojne matrice dimenzije *M* puta *N* tako što uporedo pronalazi maksimum u svakoj vrsti matrice (maksimum svake vrste pronalazi u po jednom od pokrenutih procesa-dece), a onda pronalazi maksimum tih maksimuma. U slučaju greške, ovaj program treba da ispiše poruku o grešci i vrati status -1, a u slučaju uspeha treba da ispiše pronađeni maksimum i vrati status 0. Pretpostaviti da je matrica već nekako inicijalizovana.

```
const int M = ..., N = ...;
extern int mat[M][N];
```

Rešenje:

2. (10 poena)

Neki procesor pri obradi prekida, sistemskog poziva i izuzetka prelazi na sistemski stek. Taj stek alociran je u delu memorije koju koristi kernel, a na vrh tog steka ukazuje poseban registar SSP procesora koji je dostupan samo u privilegovanom režimu.

Prilikom obrade ovih situacija, procesor ništa ne stavlja na stek, već zatečene, neizmenjene vrednosti registara PC i PSW koje je koristio prekinuti proces sačuva u posebne, za to namenjene registre, SPC i SPSW, respektivno, a vrednosti registara SP i SSP međusobno zameni (*swap*). Prilikom povratka iz prekidne rutine instrukcijom `iret` procesor radi inverznu operaciju. Procesor je RISC, sa *load/store* arhitekturom i ima 32 registra opšte namene (R0..R31).

Kernel je višenitni, a svakom toku kontrole (procesu ili niti), uključujući i niti kernela, pridružen je poseban stek koji se koristi u sistemskom režimu. Prilikom promene konteksta, kontekst procesora treba sačuvati na tom steku, dok informaciju o vrhu steka treba čuvati u polju PCB čiji je pomeraj u odnosu na početak strukture PCB označen simboličkom konstantom `offsSP`.

U kodu kernela postoji statički pokazivač `oldRunning` koji ukazuje na PCB tekućeg procesa, kao i pokazivač `newRunning` koji ukazuje na PCB procesa koji je izabran za izvršavanje.

Napisati kod funkcije `yield` koju koristi kernel kada želi da promeni kontekst (prebaci se sa izvršavanja jednog toka kontrole, `oldRunning`, na drugi, `newRunning`), na bilo kom mestu gde se za to odluči.

Rešenje:

3. (10 poena)

Više procesa proizvođača i potrošača međusobno razmenjuju znakove (`char`) preko ograničenog bafera tako što svi ti procesi dele jedan zajednički logički segment memorije veličine `sizeof(bbuffer)` koji su alocirali sistemskim pozivom *mmap* kao *bss* segment (inicijalizovan nulama pri alokaciji). Svaki od tih procesa adresu tog segmenta konvertuje u pokazivač na tip `struct bbuffer` i dalje radi operacije sa ograničenim baferom pozivajući sledeće operacije iz biblioteke `bbuf` čije su deklaracije (`bbuf.h`) date dole:

- `bbuf_init`: svaki proces koji želi da koristi bafer mora najpre da pozove ovu operaciju kako bi ona otvorila potrebne semafore; ukoliko neki od sistemskih poziva vezanih za semafore nije uspeo, sve one već otvorene semafore treba osloboditi i vratiti negativnu vrednost (greška); ukoliko je inicijalizacija uspela, treba vratiti 0;
- `bbuf_close`: svaki proces koji koristi bafer treba da pozove ovu operaciju kada završi sa korišćenjem bafera;
- `bbuf_append`, `bbuf_take`: operacije stavljanja i uzimanja elementa (`char`) u bafer.

Sve ove operacije podrazumevaju da je argument ispravan pokazivač (ne treba ga proveravati). Sve operacije osim `bbuf_init` takođe podrazumevaju da je inicijalizacija pre toga uspešno završena – proces ih ne sme pozivati ako nije, pa ne treba proveravati ispravnost stanja bafera.

Implementirati biblioteku `bbuf` (`bbuf.cc`): dati definiciju strukture `bbuffer` i implementirati sve ove operacije korišćenjem POSIX semafora.

```
struct bbuffer;
int  bbuf_init (struct bbuffer*);
void bbuf_append (struct bbuffer*, char);
char bbuf_take (struct bbuffer*);
void bbuf_close (struct bbuffer*);
```

Rešenje: