Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Treći, januar 2020.
Datum: 8. 1. 2020.

Treći kolokvijum iz Operativnih sistema 2

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 = _____%

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) Upravljanje diskovima

Klasa `DiskScheduler`, čiji je interfejs dat dole, implementira algoritam *SSTF* raspoređivanja zahteva za operacije na disku. Da bi se sprečilo izgladnjivanje, tj. predugo i neravnomerno vreme čekanja zahteva, primenjuje se sledeća tehnika. Postoji dovoljno veliki broj `NumReqQueues` različitih redova čekanja, pri čemu je svaki red uređen po redosledu potrebnom za opsluživanje po *SSTF* algoritmu. Novi zahtevi se smeštaju u red koji je trenutno aktivan za smeštanje. Svake periode *T* (reda nekoliko stotina ms) sistem poziva operaciju `DiskScheduler::period` kojom se menja aktivan red na nov red u koji se smeštaju novi zahtevi. Zahtevi se opslužuju iz reda koji je na redu za opsluživanje, sve dok taj red ne postane prazan, kada se prelazi na hronološki sledeći red. Na taj način, zahtevi se opslužuju uglavnom po *SSTF* redosledu iz tekućeg reda, ali novi zahtevi neće neograničeno stizati u red koji se opslužuje, tako da će on vremenom postati prazan i svi zahtevi iz njega biće opsluženi u ograničenom vremenu. Sa druge strane, paketi zahteva grupisani po vremenskim intervalima *T* se opslužuju približno hronološki, po FIFO redosledu.

Zahtev je predstavljen strukturom `Req`, a pomoćna klasa `ReqList` implementira listu zahteva uređenu po cilindrima na odgovarajući način, tako da operacija `put` stavlja nov zahtev u listu, a operacija `get` vraća zahtev (ako ga ima) koji je sledeći u listi na redu za opsluživanje po *SSTF* algoritmu. Ova klasa je implementirana.

Implementirati klasu `DiskScheduler` i sve njene operacije.

```
class DiskScheduler {
public:
  DiskScheduler () : in(0), out(0) {}

  Req* get ();
  void put (Req* r);
  void period ();

};

class ReqList {
public:
  ReqList ();
  Req* get ();
  void put (Req* r);
};
```

Rešenje:

2. (10 poena) Operativni sistem Linux

Napisati bash skript koji ispisuje broj fajlova u celom sistemu koji imaju ekstenziju txt i koji sadrže reč koja je prosleđena kao prvi argument skripta. U obzir uzimati samo one fajlove za koje korisnik ima pravo čitanja. U slučaju nekorektnog broja argumenata prekinuti izvršavanje skripta i prijaviti grešku.

Rešenje:

3. (10 poena) Operativni sistem Linux

Napisati program na programskom jeziku C koji ima isti efekat kao izvršavanje sledeće komande /bin/ls | /usr/bin/sort u komandnoj liniji. U rešenju koristiti neimenovane cevi operativnog sistema Linux. Na raspolaganju su poznati POSIX sistemski pozivi, kao i sledeće POSIX stvari uz koje je dat delimičan izvod iz dokumentacije:

```
#include <unistd.h>
int dup(int oldfd);
```

The dup() system call creates a copy of the file descriptor oldfd, using the lowest-numbered unused file descriptor for the new descriptor.
After a successful return, the old and new file descriptors may be used interchangeably. They refer to the same open file description (see open(2)) and thus share file offset and file status flags; for example, if the file offset is modified by using lseek(2) on one of the file descriptors, the offset is also changed for the other.

```
#include <stdio.h>
extern FILE *stdin;
extern FILE *stdout;
extern FILE *stderr;
```

Under normal circumstances every UNIX program has three streams opened for it when it starts up, one for input, one for output, and one for printing diagnostic or error messages. These are typically attached to the user's terminal (see tty(4)) but might instead refer to files or other devices, depending on what the parent process chose to set up.
...
On program startup, the integer file descriptors associated with the streams stdin, stdout, and stderr are 0, 1, and 2, respectively. The preprocessor symbols STDIN_FILENO, STDOUT_FILENO, and STDERR_FILENO are defined with these values in <unistd.h>.

Rešenje: