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: Treći, jun 2018.
Datum: 20. 6. 2018.

Treći 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 = _____% = _____/10

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) Ulaz/izlaz

Dat je neki sekvencijalni, znakovno orijentisani ulazni uređaj (ulazni tok) sa koga se znak učitava sledećom funkcijom:

```
char getchar();
```

Od ovog uređaja napraviti apstrakciju sekvencijalnog, blokovski orijentisanog ulaznog uređaja, sa koga se blok veličine `BlockSize` učitava na zadatu adresu funkcijom:

```
void readBlock(char* addr);
```

Ignorisati sve greške.

Rešenje:

2. (10 poena) Fajl sistem

Dole je dat izvod iz dokumentacije za API za fajlove u GNU sistemima. Date deklaracije su u `<unistd.h>` i `<fctl.h>`. Korišćenjem samo dole datih funkcija, realizovati sledeću funkciju koja prepisuje ceo sadržaj datog ulaznog fajla proizvoljne veličine u dati izlazni fajl, korišćenjem svog bafera određene veličine. U slučaju bilo kakve greške, funkcija treba da vrati negativnu vrednost, u suprotnom treba da vrati 0.

```
int fcopy (const char *filenamefrom, const char *filenameto);
```

`int open (const char *filename, int flags)`
Creates and returns a new file descriptor for the file named by `filename`. Initially, the file position indicator for the file is at the beginning of the file. The `flags` argument controls how the file is to be opened. This is a bit mask; you create the value by the bitwise OR of the appropriate parameters (using the '|' operator in C) – the following macros:

`O_RDONLY`	*Open the file for read access.*
`O_WRONLY`	*Open the file for write access.*
`O_RDWR`	*Open the file for both reading and writing.*
`O_CREAT`	*If set, the file will be created if it doesn't already exist.*
`O_APPEND`	*The bit that enables append mode for the file. If set, then all write operations write the data at the end of the file, extending it, regardless of the current file position.*
`O_TRUNC`	*Truncate the file to zero length.*

The normal return value from `open` is a non-negative integer file descriptor. In the case of an error, a value of -1 is returned instead.

`int close (int filedes);`
Closes the file descriptor `filedes`. The normal return value is 0. In the case of an error, a value of -1 is returned.

`ssize_t`
This data type is used to represent the sizes of blocks that can be read or written in a single operation. It is similar to `size_t`, but must be a signed type.

`ssize_t read (int filedes, void *buffer, size_t size);`
Reads up to `size` bytes from the file with descriptor `filedes`, storing the results in the `buffer`. (This is not necessarily a character string, and no terminating null character is added.)
The return value is the number of bytes actually read. This might be less than `size`; for example, if there aren't that many bytes left in the file or if there aren't that many bytes immediately available. The exact behavior depends on what kind of file it is. Note that reading less than size bytes is not an error.
A value of zero indicates end-of-file (except if the value of the `size` argument is also zero). This is not considered an error. If you keep calling read while at end-of-file, it will keep returning zero and doing nothing else.
If `read` returns at least one character, there is no way you can tell whether end-of-file was reached. But if you did reach the end, the next read will return zero. In case of an error, `read` returns -1.

`ssize_t write (int filedes, const void *buffer, size_t size);`
Writes up to `size` bytes from `buffer` to the file with descriptor `filedes`. The data in `buffer` is not necessarily a character string and a null character is output like any other character.
The return value is the number of bytes actually written. This may be `size`, but can always be smaller. Your program should always call write in a loop, iterating until all the data is written. In the case of an error, `write` returns -1.

Rešenje:

3. (10 poena) Fajl sistem

Neki fajl sistem koristi indeksiranu alokaciju sadržaja fajla sa jednostepenim indeksom u jednom bloku. Struktura FCB keširana je u memoriji, kao i indeksni blok. U strukturi FCB, pored ostalih, postoje sledeća polja:

- `unsigned long size`: veličina sadržaja fajla u bajtovima;
- `unsigned long* index`: pokazivač na (keširan) indeks (ulazi su tipa `unsigned long`).

Osim toga, definisane su i sledeće konstante i funkcija:

- `unsigned long BlockSize`: veličina bloka na disku u bajtovima;
- `unsigned long MaxFileSize`: maksimalna dozvoljena veličina sadržaja fajla u bajtovima; jednaka je maksimalnom broju ulaza u indeksu pomnoženom veličinom bloka `BlockSize`;
- `unsigned long allocateBlock()`: alocira jedan slobodan blok na disku i vraća njegov broj; ukoliko slobodnog bloka nema, vraća 0.

Realizovati funkciju

```
unsigned long extend (FCB* fcb, unsigned extension);
```

koja proširuje sadržaj fajla na čiji FCB ukazuje prvi argument za broj bajtova dat drugim argumentom. Ukoliko traženo proširenje premašuje maksimalnu veličinu fajla ili na disku nema dovoljno slobodnog prostora, sadržaj fajla treba proširiti koliko je moguće. Ova funkcija vraća broj bajtova sa koliko je stvarno uspela da proširi sadržaj (jednako ili manje od traženog).

Rešenje: