Lab-2. Dretve

Višedretvenost omogućava bolju iskorištenost računalnog sustava paralelnim radom dretvi istog ili različitih procesa. Tako se mogu iskoristiti višeprocesorski sustavi, ali i razni drugi elementi računala (disk, mreža, ulazno-izlazne naprave).

U nastavku je opisano stvaranje dretvi prema POSIX sučelju.

Stvaranje dretve obavlja se funkcijom `pthread_create` koja prima nekoliko argumenata:

1. Prvi argument je kazaljka na element tipa `pthread_t` – „opisnik dretve“
2. Drugi argument se može koristiti da se detaljnije definiraju parametri dretve, primjerice, stog, način raspoređivanja i slično. Postavljanjem vrijednosti `NULL` koriste se uobičajene vrijednosti.
3. Treći argument je kazaljka na funkciju koja će biti početna funkcija nove dretve.
4. Četvrti argument je kazaljka koja će se poslati kao argument početne funkcije dretve.

Primjer jednostavnog programa koji stvara nekoliko dretvi te potom čeka da one završe s radom je u nastavku.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

#define BROJ_DRETVI 3
#define BROJ_ITERACIJA 5

void *dretva (void *rbr)
{
    int i, *d = rbr;
    for (i = 1; i <= BROJ_ITERACIJA; i++) {
        printf("Dretva %d; iteracija=%d\n", *d, i);
        sleep(1);
    }
    return NULL;
}
int main(int argc, char *argv[])
{
    pthread_t opisnik[BROJ_DRETVI];
    int i, j, id[BROJ_DRETVI];

    for (i = 0; i < BROJ_DRETVI; i++) {
        id[i] = i;
        if (pthread_create(&opisnik[i], NULL, dretva, &id[i])) {
            fprintf(stderr, "Ne mogu stvoriti novu dretvu!\n");
            exit(1);
        }
    }

    for (j = 0; j < BROJ_DRETVI; j++)
        pthread_join(opisnik[j], NULL);

    return 0;
}
```

Stvorena dretva radi dok ne završi sa zadanom funkcijom – dok ne izađe iz nje, ili to može napraviti i prije pozivom `pthread_exit()`.

Početna dretva nakon stvaranja novih, može preko `pthread_join()` čekati na završetak stvorene dretve.

Zadatak

Nadograditi program iz prve vježbe (ali u lab2) tako da se koristi više dretvi (barem tri ukupno). Barem jedna dretva neka računa isto kao i u prvoj vježbi, dok druga (ili više njih) na neki način utječe na rad prve. Primjerice, jedna dretva čita naredbe preko konzole (npr. scanf), druga čeka na naredbe preko „mreže“.

Naredba u ovom kontekstu može biti bilo što. Za primjer zadatka iz prve vježbe to primjerice može biti samo promjena trenutnog broja. Npr. preko konzole se upiše 25 pa je idući broj koji će koristiti dretve u obradi taj (tj. idući 26).

„Mreža“ se ne mora izravno koristiti već primjerice preko cjevovoda (kako je predloženo u predlošku rješenja). Imenovani cjevovod napraviti s naredbom *mkfifo ime-cijevi*. Cjevovod radi kao i datoteka – iz programa se koriste ista sučelja. Međutim, ona „ne postoji“ dok ju obje strane ne otvore, jedna za pisanje a druga za čitanje. Isprobati princip rada naredbama: *cat ime-cijevi* i *echo test > ime-cijevi* pokrenutih iz različitih konzola proizvoljnim redoslijedom.

Koristiti *Makefile* za prevođenje i pokretanje uz ciljeve *pokreni* i *obrisi*. Izvorni kod podijeliti u više datoteka, primjerice kako je predloženo (lab2.c, datoteke.c/h, signali.c/h).