## *Primjer s višedretvenosti*

Ostvariti podsustav za višedretvenost sa sučeljem:

```
  void inicijaliziraj_i_pokreni_visedretvenost();
  dretva_t *stvori_dretvu(void *pocetna_funkcija, void *argument);
  dretva_t *dohvati_aktivnu();
  void dodaj_u_pripravne(dretva_t *dretva);
  void dodaj_na_kraj_reda(dretva_t *dretva, red_dretvi_t *red);
  dretva_t *dohvati_prvu_iz_reda(red_dretvi_t *red);
  dretva_t *uzmi_prvu_iz_reda(red_dretvi_t *red);
  void rasporedjivac();
```

Dostupno:

```
  void *kmalloc(size_t size);

  void stvori_pocetni_kontekst_dretve_na_stogu(dretva_t *dretva, void
  *pocetna_funkcija, void *argument, void *stog, size_t velicina_stoga);

  void inicijaliziraj_praznu_listu(lista_t *lista);
  void dodaj_na_početak_liste(lista_t *lista, element_liste *e);
  void dodaj_na_kraj_liste(lista_t *lista, element_liste *e);
  void dodaj_u_sortiranu_listu(lista_t *lista, element_liste *e,
     int (*usporedi)(element_liste *e1, element_liste *e2));
  element_liste *dohvati_prvi_element_liste(lista_t *lista);
  element_liste *uzmi_prvi_element_liste(lista_t *lista);
  element_liste *uzmi_element_liste(lista_t *lista, element_liste *e);

  void prebaci_se_na_dretvu(dretva_t *stara, dretva_t *nova);
```

Pretpostaviti da su svi argumenti OK - ne treba ih provjeravati. Raspoređivanje ostvariti prema prioritetu. Definirati potrebne strukture podataka: `dretva_t` i `red_dretvi_t`. Strukture `lista_t` i `element_liste` su definirani i ne koriste se izravno iz ova koda.

*Rješenje*

```
typedef struct _dretva_t_ {
      int id;
      void *kontekst;
      int stanje;
      int prio;
      element_liste element;
}
dretva_t;

typedef lista_t red_dretvi_t;

dretva_t *dohvati_opisnik_dretve(element_liste *e) {
      return container_of(e, dretva_t, element);
}

static dretva_t *aktivna;
static red_dretvi_t pripravne;
```

```
static void latentna_dretva(void *x) {
      while (1) ;
}
void inicijaliziraj_i_pokreni_visedretvenost() {
      aktivna = NULL;
      inicijaliziraj_praznu_listu(&pripravne);
      stvori_dretvu(latentna_dretva, NULL);
      rasporedjivac();
}
dretva_t *stvori_dretvu(void *pocetna_funkcija, void *argument){
      dretva_t *dretva = kmalloc(sizeof(dretva_t));
      void *stog = kmalloc(VELICINA_STOGA);
      stvori_pocetni_kontekst_dretve_na_stogu(dretva, pocetna_funkcija,
            argument, stog, VELICINA_STOGA);
      dodaj_u_pripravne(dretva);
}
dretva_t *dohvati_aktivnu() {
      return aktivna;
}
static int usporedi(element_liste *e1, element_liste *e2){
      dretva_t *dretva1 = dohvati_opisnik_dretve(e1);
      dretva_t *dretva2 = dohvati_opisnik_dretve(e2);
      return dretva1->prio - dretva2->prio;
}
void dodaj_u_pripravne(dretva_t *dretva) {
      dodaj_u_sortiranu_listu(&pripravne, &dretva->element, usporedi);
      dretva->stanje = PRIPRAVNA;
}
void dodaj_na_kraj_reda(dretva_t *dretva, red_dretvi_t *red) {
      dodaj_na_kraj_liste(red, &dretva->element);
}
dretva_t *dohvati_prvu_iz_reda(red_dretvi_t *red){
      return dohvati_opisnik_dretve(dohvati_prvi_element_liste(red));
}
dretva_t *uzmi_prvu_iz_reda(red_dretvi_t *red){
      return dohvati_opisnik_dretve(uzmi_prvi_element_liste(red));
}
void rasporedjivac() {
      dretva_t *prije = aktivna;
      dretva_t *poslije = dohvati_prvu_iz_reda(&pripravne);
      assert(prije || poslije);
      if (prije == NULL || (poslije && prije->prio < poslije->prio) {
            aktivna = uzmi_prvu_iz_reda(&pripravne));
            aktivna->stanje = PRIPRAVNA;
            if (prije && prije->stanje == AKTIVNA)
                  dodaj_u_pripravne(prije);
            prebaci_se_na_dretvu(prije, aktivna);
      }
}
```