

# Operativni Sistemi

---

VEŽBE 03 – NITI

# Konkurentnost

---

- Nastao je kao mehanizam za optimizaciju upotrebe procesora. Izuzetno je aktuelan zato što omogućava:
- Jednostavniju realizaciju funkcionalno nezavisnih delova programa (npr. glavni program i pozadinske funkcije)
- Bolje i ravnomernije iskorišćenje procesora (umesto da procesor čeka neki proces, može da izvršava neki drugi proces za to vreme)
- **Potencijalno** ubrzanje izvršavanje programa korišćenjem više jezgara/procesora (umesto da jedan procesor sabira milion elemenata, bolje da 4 procesora sabiraju po 250000 elemenata)

# Konkurentno programiranje

---

- Na ovim vežbama ćemo govoriti o konkurentnom programiranju upotrebom:
- Niti i
- Kritičnih sekcija

# Niti (threads)

---

U operativnom sistemu niti predstavljaju:

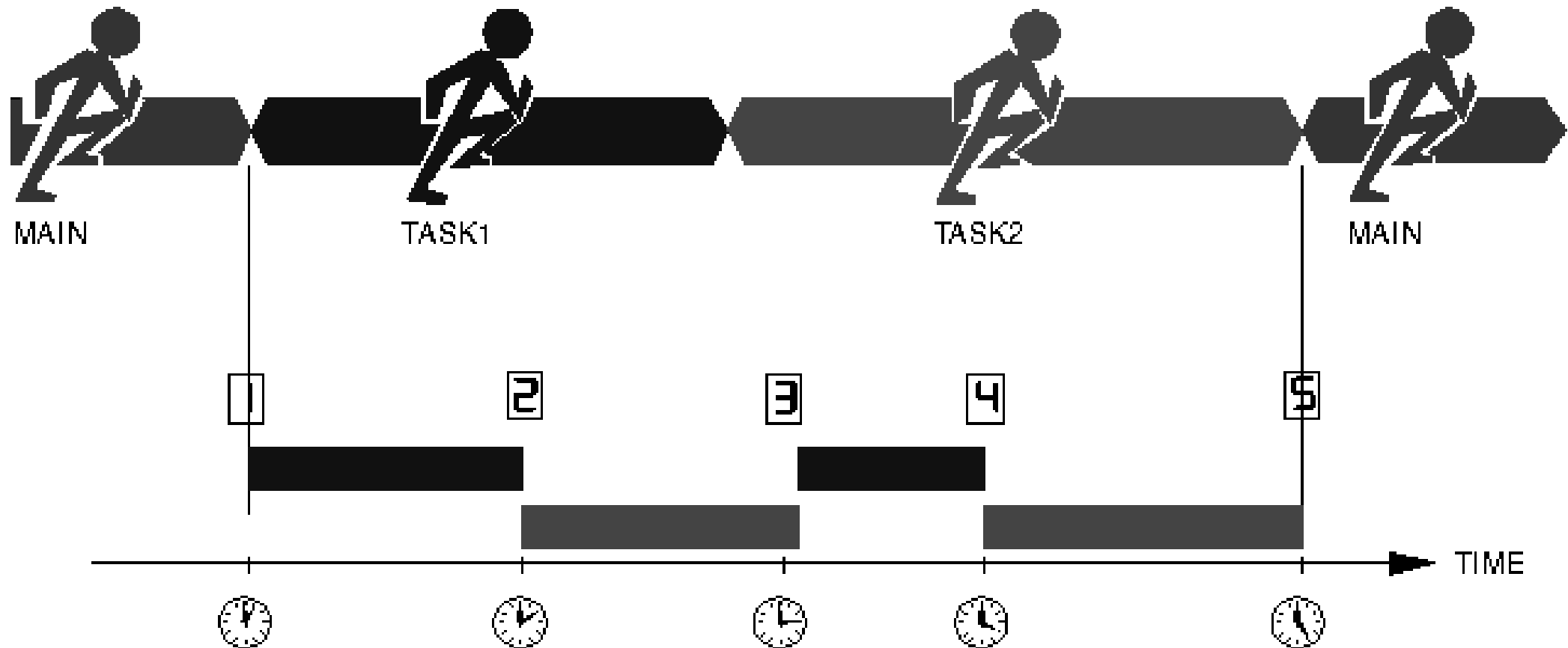
- **tokove izvršavanja, tj. funkcije programskog koda** (koji se prepliću na procesoru)
- čine osnovu raspoređivanja (*scheduling entity*)

# Sekvencijalno i konkurentno izvršavanje

---

- Svaki program ima bar jednu nit (tok izvršavanja) nastalu od funkcije `main()`.
- Program koji se sastoji od samo jedne niti se naziva sekvencijalni program.
- Program koji se sastoji od više (od jedne) niti se naziva konkurentni program.

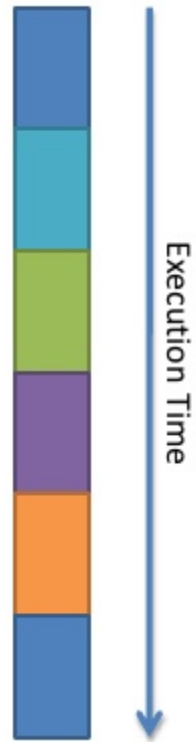
# Sekvencijalno i konkurentno izvršavanje u jednoprocorskom sistemu



# Sekvencijalno i konkurentno izvršavanje u višeprocorskom sistemu

---

Sequential



Concurrent



# Od čega nastaju niti?

---

- Niti nastaju od funkcija
- Svaka niti kada se kreira krene da izvršava „svoj“ kod tj. telo neke funkcije koja je prosleđena konstruktoru niti.



# Funkcija main

---

- Ulazna tačka (entry point) programa tj. početak korisničkog dela programa je funkcija `main()`
- Završetak korisničkog dela programa nastupa kada se završi funkcija `main()`
- Ovo važi kako za sekvencijalni tako i za konkurentni program! iz čega sledi...

# Nit nastala od funkcije main()

---

- Ova nit je naročito bitna.
- Program traje koliko i main()
- Kada se završi main(), završava se i ceo program.
- Osim toga, main niti je kao i sve druge niti.

# Objekat klase thread (nit)

---

- Služi za stvaranje niti (toka izvršavanja).
- Kada se stvori nit objekat klase thread je u stanju 'joinable' (pokazuje na tok izvršavanja). Ovo znači da je nit krenula da se izvršava.
- se prevodi iz stanja 'joinable' operacijama:
  - join()
  - detach().
- Ukoliko se nit ne prevede iz stanja 'joinable' nekom od dve metode (join ili detach) dobiće se greška: **”terminate called without an active exception. Core dumped”**.

# Razlika između join() i detach()

---

- Operacija join() blokira nit pozivaoca dok se nit na kojoj je operacija join pozvana ne završi
- Koristi se kada nit main čeka rezultat rada niti koje je stvorio
- Operacija detach() razdvaja nit pozivaoca od niti na kojoj je operacija detach pozvana, tako da nit pozivaoc ne čeka da se nit na kojoj je operacija detach pozvana završi
- Koristi se kada postoje ciklične niti (daemon) koje izvršavaju neku funkciju u beskonačnoj petlji. Nit main tada uz pomoć detach može da završi rad, a u suprotnom bi čekala beskonačno

# Argumenti i povratna vrednost

---

- Ako je nit nastala od funkcije `f()`:
  - Pri stvaranju, niti se moraju proslediti argumenti po istim pravilima kao da se poziva obična C funkcija `f()`
  - Vrednosti svih argumenata niti se pri stvaranju kopiraju u kontekst niti
  - Povratna vrednost funkcije `f()` se zanemaruje (uvek je `void`)