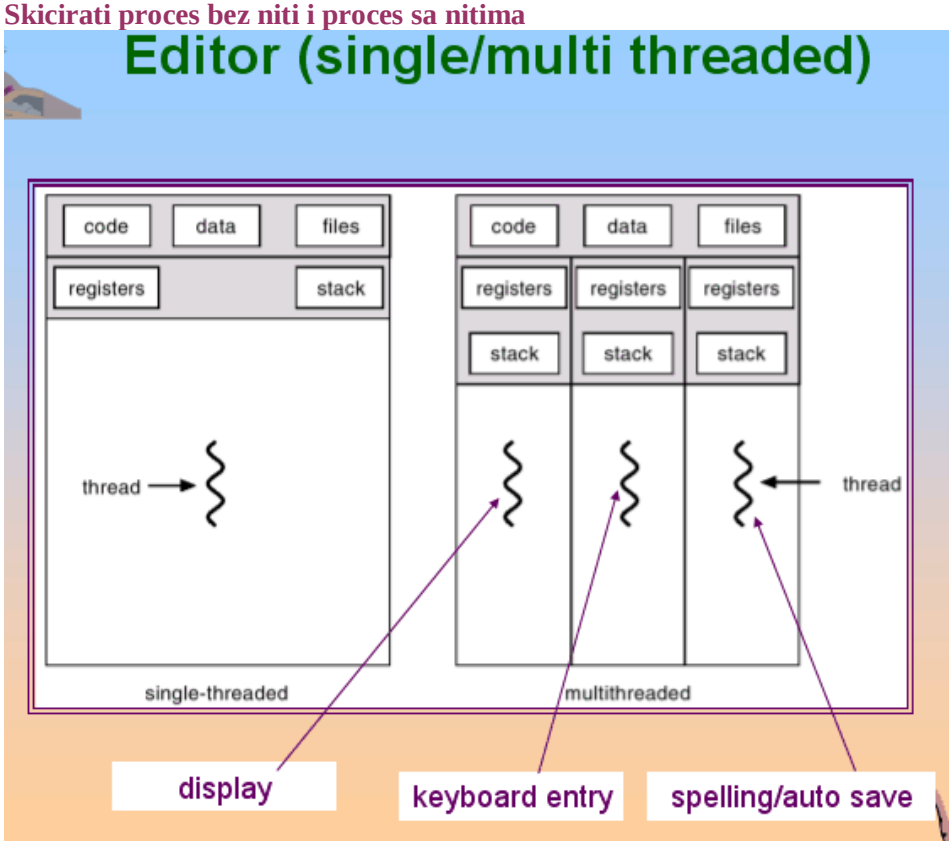
# Uvod u niti

1. **Objasniti pojam niti u konteksu procesa. Razlozi za uvođenje niti, prednosti**

   **Niti (engl. *threads*), to jest laki procesi (engl. *lightweight processes, LWP*), predstavljaju osnovne celine za izvršavanje koda pod savremenim operativnim sistemima. Niti su programska celina koja treba da obavi jedan posao. Niti (jedna ili više) pripadaju jednom klasičnom, tj. teškom (engl. *heavzweight*) procesu. U klasičnom kontekstu, jedan težak proces ima svoj programski brojač i druge procesorske registre, tri memorijske sekcije (kod, podaci i stek) i I/O resurse kao što su datoteke i uređaji.**

   **Dobre osobine :**
   - **Vreme odziva *(Responsiveness)***
   - **Deljenje resursa *(Resource Sharing)***
   - **Ekonomičnost**
   - **Bolje iskorišćenje višeprocesorske arhitekture**

2. **Skicirati proces bez niti i proces sa nitima**




3. **Navedite nekoliko prednosti multithreading-a**

   **Prednosti višenitnog procesa :**
   - **Vreme odziva *(Responsiveness)***
   - **Deljenje resursa *(Resource Sharing)***
   - **Ekonomičnost**
   - **Bolje iskorišćenje višeprocesorske arhitekture**

4. **Šta je zajedničko za niti unitar jenog procesa, a šta je unikatno**

   **Niti kao laki procesi i delovi jednog istog procesa, imaju svoje unikatne resurse i zajedničke resurse sa ostalim nitima istog procesa. Od unikatnih resursa imaju poseban identifikator niti (engl. *thread ID*), posebnu vrednost programskog brojača vrednosti drugih registra procesora, i poseban stek. Sve ostalo (sekcija koda, sekcija podataka, otvorene datoteke, signali) zajednički su resursi za sve niti jednog procesa.**

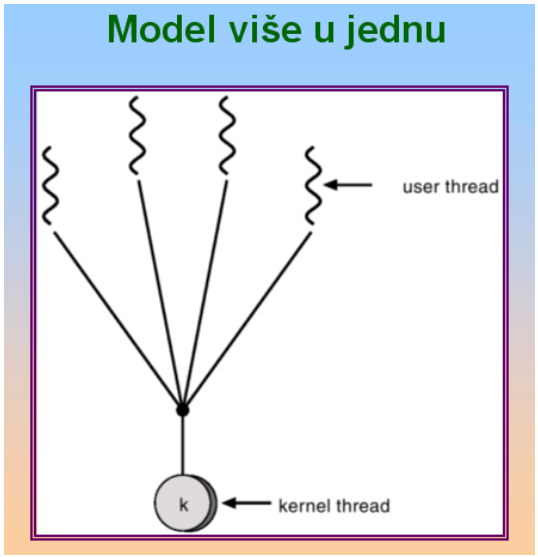
## 5. Diskutovati kernelske i korisničke niti?

**Kako jedan proces može da se izvršava i u korisničkom režimu i u režimu jezgra (engl. *kernel*), potrebno je obezbediti istu podršku i na korisničkom nivou za korisničke niti (engl. *user threads*), i na nivou jezgra za niti jezgra (engl. *kernel threads*).**
**Podrška za korisničke niti realizuje se preko biblioteke za rad sa korisničkim nitima (engl. *user thread library*). Ova biblioteka obezbeđuje podršku za stvaranje niti, raspored izvršavanja niti i upravljanje nitima, ali bez uticaja jezgra(*kernel*). Najznačajnije vrste niti su : POSIX Pthreads, Mach C-threads i Solaris UI-threads.**
**Niti jezgra (*kernel*) direktno podržava operativni sistem. Konkretno, jezgro (*kernel*) izvršava operacije stvaranja niti, raspoređivanja niti i upravljanja nitima u prostoru jezgra. Primeri : Windows 95/98/NT/2000, Solaris, Tru64 UNIX , BeOS, Linux.**

## 6. Koji multithread modeli postoje?

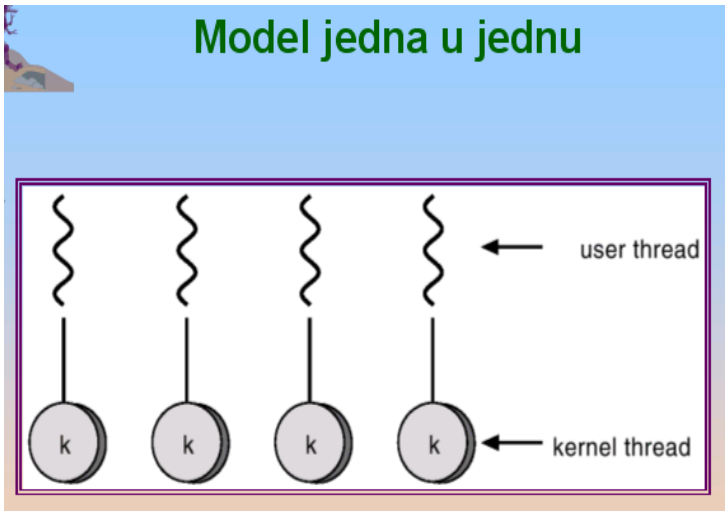
**Višenitni modeli:**

- **Više u jednu *(Many-to-One)***
- **Jedna u jednu *(One-to-One)***
- **Više u više *(Many-to-Many)***



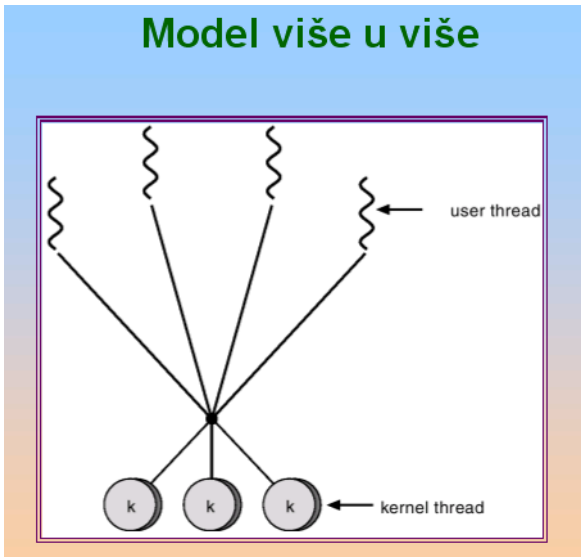

## 7. Objasniti many-to-one model (skicirati)

**U ovom modelu, više korisničkih niti mapira se u jednu nit jezgra. Upravljanje nitima se odvija na korisničkom nivou i može biti efikasno, ali ima nedostatke. Ako neka od niti obavi bilo koji blokirajući sistemski poziv, blokiraće se ceo proces, to jest sve njegove niti. S obzirom na to da samo jedna nit može pristupiti jezgru u jednom trenutku, u režimu jezgra neće moći da se iskoristi višeprocesorska arhitektura. Ovaj model koristi operativni sistem Solaris i biblioteku Green threads.**




## 8. Objasniti one-to-one model (skicirati)

**U ovom modelu, koji je karakterističan za operativne sisteme Windows NT, Windows 2000/XP/2003 i OS/2, svaka korisnička nit mapira se u jednu nit jezgra. Ovaj model obezbeđuje mnogo bolje konkurentno izvršavanje niti, dozvoljavajući da druge niti nastave aktivnosti u slučaju kada jedna nit obavi blokirajući sistemski poziv. Takođe se omogućava da se više niti jezgra izvršavaju paralelno na višeprocesorskoj arhitekturi. Kako se mnogo vremena gubi pri stvaranju i održavanju niti jezgra, mnogi operativni sistemi ograničavaju maksimalan broj niti jezgra.**




## 9. Objasniti many-to-one model (skicirati)

**U ovom modelu, više korisničkih niti mapira se u manji ili isti broj niti jezgra, pri čemu mapiranje zavisi od operativnog sistema a naročito od broja procesora. Ovo je najkompleksniji i najkvalitetniji model jer može da se prilagodi i aplikaciji i procesorskom okruženju. Ovaj više nitni model zastupljen je u sistemima Solaris-2, IRIX, HP-UX i True64 UNIX.**

## 10. Koje su promene kod kreiranja procesa, prekidanja procesa i IPC za multihread?

**U više nitnoj arhitekturi menja se simantika sistemskih poziva za upravljanje procesa i rad s njima. Mora postojati poziv za uravljanje teškog procesa, koji će duplicirati sve niti i njihove adresne prostore, kao i poziv za pravljenje jedne jedine niti.**
**Prekidanje niti je aktivnost koja prekida izvršenje niti pre njenog prirodnog završetka. Nit koja će biti prekinuta obično se naziva ciljna nit (engl. *target thread*). Prekidanje ciljne niti može se dogoditi po dva scenarija:**

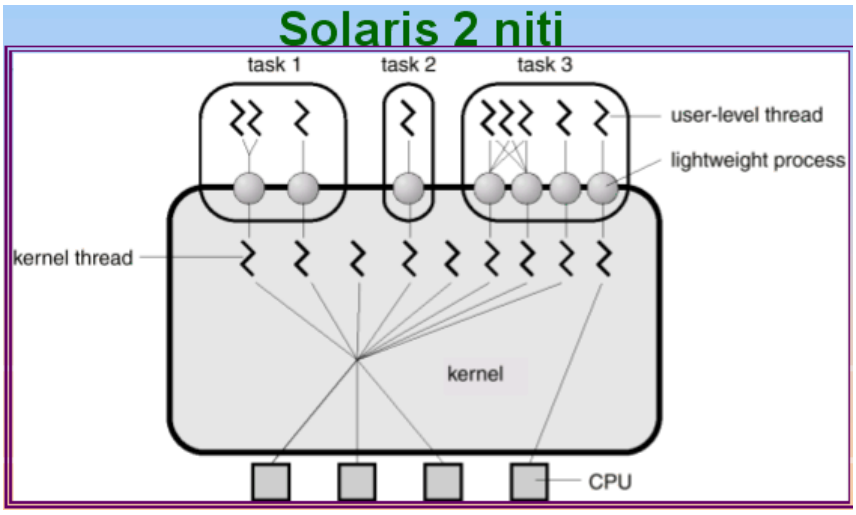
- **asinhroni prekid (engl. *asynchronous cancellation*): aktivnost niti se odmah prekida bez obzira na stanje u kome se nalazi;**
- **odloženi prekid (engl. *deferred cancellation*): ciljna nit periodično proverava treba li da prekine aktivnost; ako treba, to će učiniti u povoljnom trenutku – kad završi neku celinu, tj. obavi deo posla do kraja.**

## 11. Pthreads?

- **POSIX standard (IEEE 1003.1c) = pravila za**
  - **kreiranje niti**
  - **njihovu sinhronizaciju.**
- **Postoji API i Pthread biblioteka na korisničkom nivou, koja programeru omogućava kreiranje niti .**
- **Prihvaćeni su u UNIX operativnim sistemima.**

## 12. Objasniti Solaris Threads LWP

**Solaris 2 je verzija UNIX sistema koja podržava i korisničke i kernel niti. Solaris 2 podržava Phtreads API za korisničke niti. Solaris 2 definiše međunivo za niti koji spaja user niti i kernel niti (model više u više) na sledeći način: između user niti i kernel niti nalazi se laki proces (LWP) koji spaja, pri čemu svakom LWP-u odgovara tačno jedna kernel nit.**



## 13. Linux threads?

**Linux sistemi obezbeđuju tehnike za rad sa nitima, počev od kernel-a v2.2. Umesto termina nit, Linux koristi termin posao (engl. *task*). Osim sistemskog poziva fork, koji služi za pravljenje novih procesa, postoji i sistemski poziv clone koji formira Linux niti. I fork clone prave decu procese od roditeljskih procesa. Osnovna razlika između ovih sistemskih poziva leži u adresnom prostoru novih procesa koje prave. Sistemski poziv fork pravi proces dete kopiju adresnog prostora roditeljskog procesa, dok u sistemskom pozivu clone dva procesa postaju niti koje dele isti adresni prostor.**

## 14. Windows 2000 threads?

- **Primenjuju jedna u jednu mapiranje.**
- **Svaka nit sadrži**
  **- nit id**
  **- registarski set**
  **- korisnički stek i kernelski stek**
  **- mesto za privatne poruke**