**3. Niti (threads)**

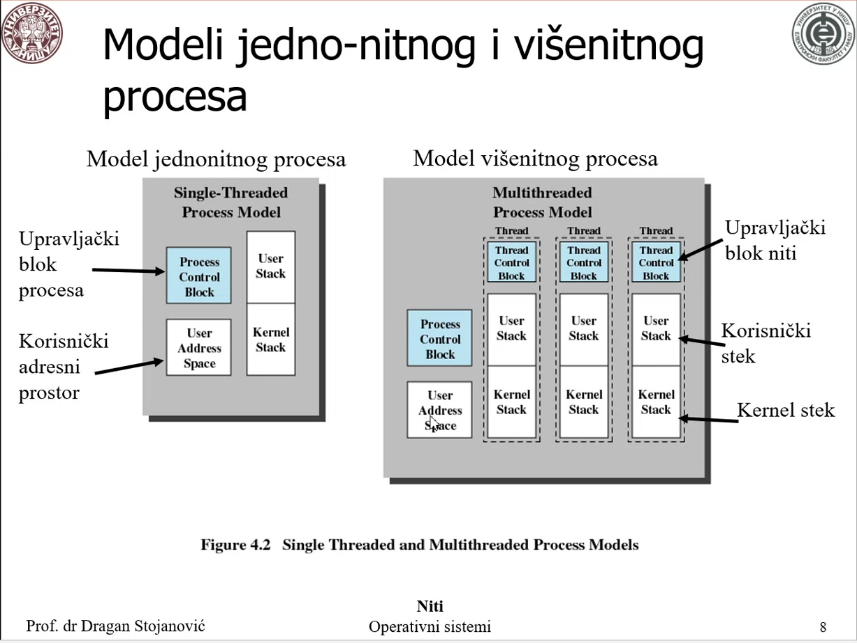
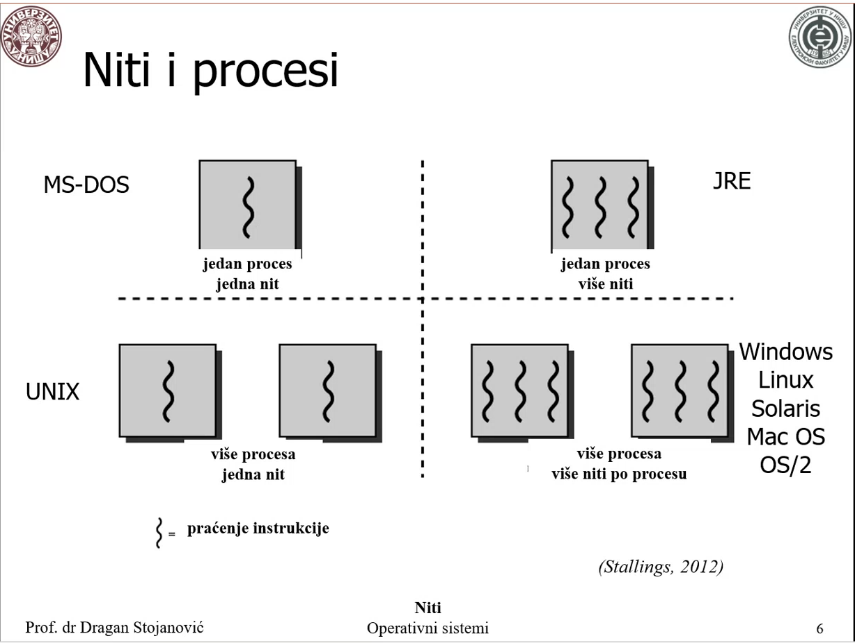
**Procesi i niti**

* Koncept procesa obuhvata dve karakteristike:
  1. **Vlasništvo nad resursom**
     + Proces sadrži virtuelni adresni proctor u kome smešta sliku procesa.
       - Slika procesa: program, podaci, stek, atributi definisani u PCB (Process Control Block)
     + Povremeno procesu može biti dodeljeno upravljanje ili vlasništvo nad esursima (glavna memorija, UI kanali, UI uređaji, datoteke)
     + OS ima funkciju zaštite da ne bi došlo do neželjenog mešanja procesa
  2. **Raspoređivanje/izvršavanje**
     + Izvršavanje procesa prati putanju izvršenja kroz jedan ili više programa (trace)
       - Ovo izvršavanje može biti isprepletano sa drugim procesima
     + OS raspoređuje procese, dodeljuje im resurse i startuje izvršenje
  3. OVE DVE KARAKTERISTIKE OS TRETIRA NEZAVISNO!

**Niti (threads)**

* **Proces (task) predstavlja osnovnu jedinicu za dodelu, vlasništvo i zaštitu u pristupu resursima.**
* **Nit (thread, lightweight process) predstavlja entitet koji se raspoređuje I izvršava na CPU.**
* U tradicionalnom OS-u, svaki proces ima:
  1. Sopstveni adresni prostor
  2. Jednu nit izvršenja
* U modernim OS-ima postoji jedan proces koji može imati VIŠE NITI IZVRŠENJA koje se izvode konkuretntno ili paralelno.
  1. Sve niti pridružene jednom procesu dele program i resurse tog procesa
  2. Svaka nit ima sopstveni magacin i stanje
* Nti omogućava podelu posla koji je dodeljen procesu na više niti izvršenja, tako da svaka nit preuzima deo tog posla.

**Višenitnost (Multithreading)**

* **Višenitnost je sposobnost OS-a da podrži više niti izvršenja unutar jednog procesa.**
* Tradicionalni pristup – jedna nit po procesu – naziva se pristup sa jednom niti (single threaded)
* Niti u OS-ima:
  1. MS-DOS podržava jednu nit
  2. Tradicionalni UNIX podržava više procesa, ali samo jednu nit po procesu
  3. JRE je primer jednog procesa sa više niti
  4. Windoes, Solaris, Linux, Mac OS, OS/2 – podržavaju više niti

|  |  |
| --- | --- |
| **Proces** | **Nit** |
| * Svakom procesu je dodeljen:   1. Virtuelni adresni prostor gde se nalazi slika procesa – program, podaci, stek, kao i atributi definisani u upravljačkom bloku procesa (PCB)   2. Zaštićeni pristup procesoru, drugim procesima (IPC), datotekama i UI resursima (uređaji i kanali) * Unutar procesa može postojati jedna ili više niti. | * Svaka nit poseduje:   1. Stanje izvršenja niti (Izvršava se, Spemna, itd)   2. Sačuvani kontekst niti kada se ne izvršava (programski brojač, registri procesora, stek pointeri)   3. Stek (magacin) izvršavanja   4. Statičku memoriju za lokalne promenljive niti   5. Pristup memoriji i resursima svog procesa koje deli sa svim ostalim nitima. |

**Koristi od niti**

* Osnovne koristi niti se zasnivaju na dobitku u performansama:

1. Potrebno je mnogo manje vremena da bi se kreirala nova nit u postojećem procesu, nego da se kreira novi proces.
2. Potrebno je mnogo manje vremena za prekid niti nego prekid procesa.
3. Potrebno je mnogo manje vremena za prebacivanje (switch) između dve niti unutar istog procesa, nego za komutiranje procesa.
4. Niti mogu komunicirati međusobno, jer dele adresni prostor procesa i otvorene datoteke
   * Bez poziva OS kernela.

**Korišćenje niti u jednokorisničkom multiprogramskom sistemu**

* Čeoni (foreground) i pozadinski (background) poslovi
  1. Program za tabelarna izračunavanja – jedna nit prikazuje GUI i prihvata korisnički unos, dok druga nit izvršava korisničke komande i ažurira tabele
* Asinhrona obrada
  1. Autosave u programu za obradu teksta
* Brzina izvršavanja
  1. Više niti može da se izvršava konkurentno/paraleno, dok je jedna nit klobirana na U/I uređaju, druga može da se izvršava
* Modularna struktura programa

**Niti**

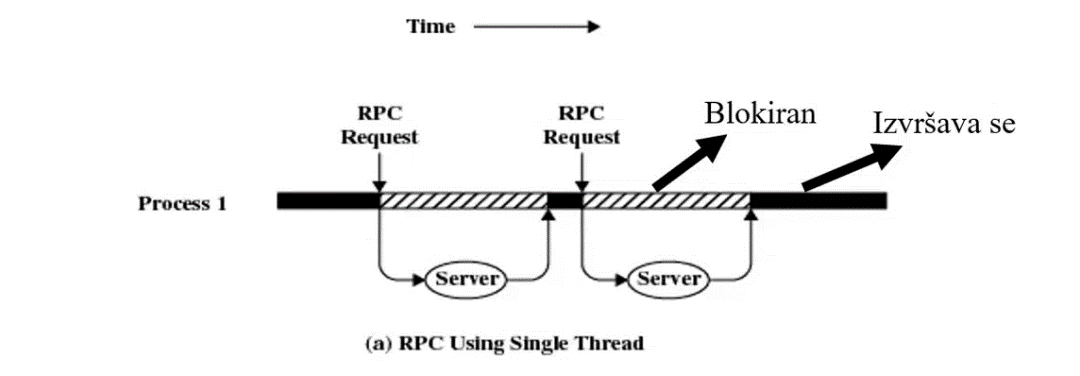
* Neke akcije u okviru OS odnose se na sve niti jednog procesa
  1. OS mora da upravlja ovim akcijama na nivou procesa.
* Primeri:
  1. Suspendovanje procesa uključuje suspendovanje svih niti procesa
  2. Terminiranje procesa terminira sve niti u okviru procesa
* Slično procesima
  1. Niti poseduju stanja izvršenja i mogu se sinhronizovati međusobno.

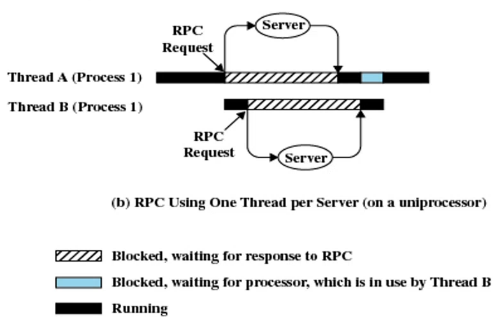
**Stanja niti**

* Osnovna stanja niti:
  1. **IZVRŠAVA SE (RUNNING)**
  2. **SPREMNA (READY)**
  3. **BLOKIRANA (BLOCKED)**
* Prelazi između stanja su isti kao kod procesa
* Ne postoje suspendovanja stanja, jer sve niti unutar istog procesa dele isti adresni prostor
  1. Sspendovanje procesa povlači suspendovanje svih niti tog procesa.
* Terminiranje procesa, terminiraju se sve niti tog procesa.

**Operacije nad nitima**

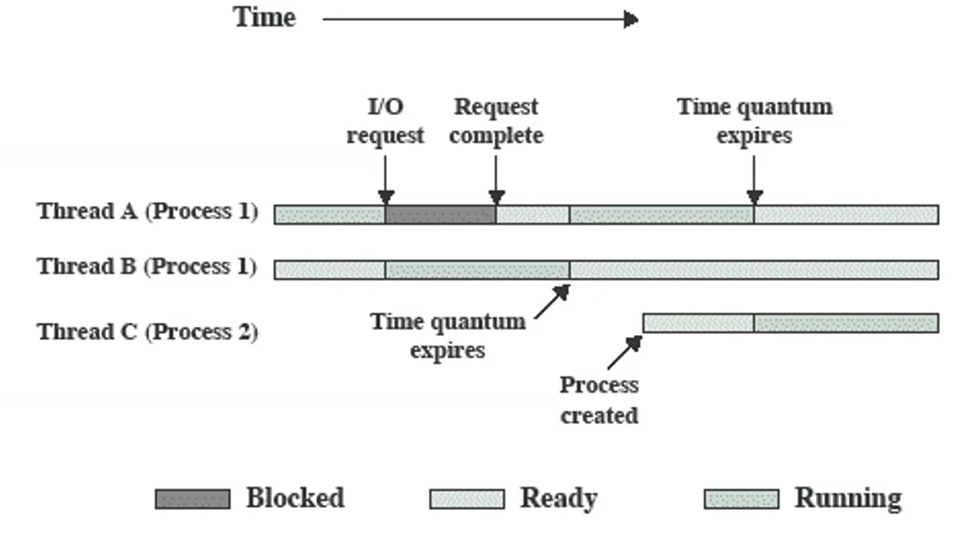
* **Kreiranje (umnožavanje, spawn)**
  1. Kreiranjem procesa kreira se nit tog procesa. Svaka nit procesa može kreirati novu nit u okviru tog procesa, obezbeđujući pointer na funkciju (instrukcije) i argumente nove niti. Nova nit dobija sopsveni kontekst (TCB), magacine i smešta se u red spremnih niti.
* **Blokiranje**
  1. Kada nit čeka na događaj (npr. U/I) blokira se, pri čemu se čuvaju vrednosti korisničkih registara, programskog brojača, stek pointera i procesor započinje izvršavanje sledeće spremne niti u istom ili drugom procesu.
* **Deblokiranje**
  1. Kada nastane događaj koji se čeka, nit se prevodi u stanje spremna
* **Završetak**
  1. Kada se nit završi, dealociraju se TCB i magacini.

**Primer: RPC korišćenjem jedne niti**

* RPC je tehnika kojom dva procesa koja se izvršavaju na različitim računarima. Međusobno komuniciraju koristeći mehanizam poziva procedure
* Program vrši dva poziva udaljenih procedura (Remote Procedure Call – RPC) na dva računara.

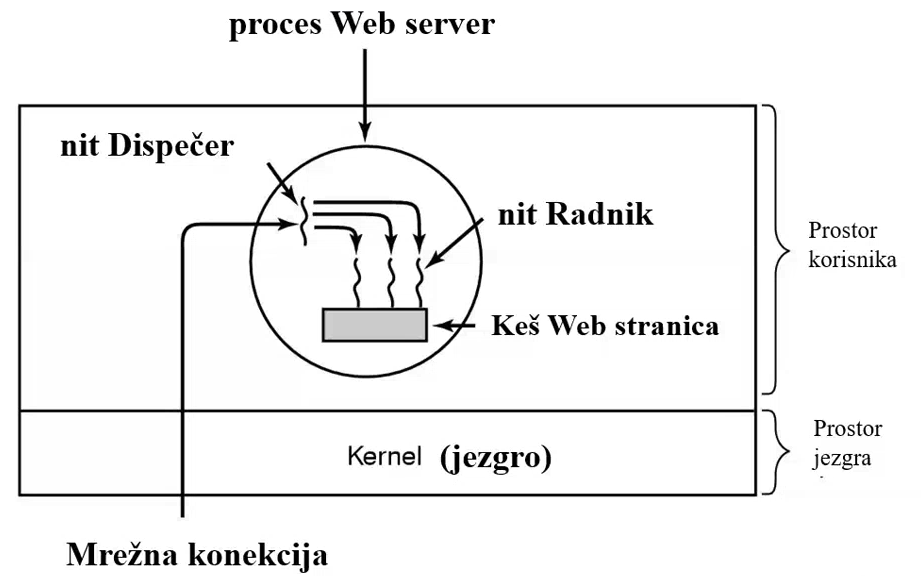
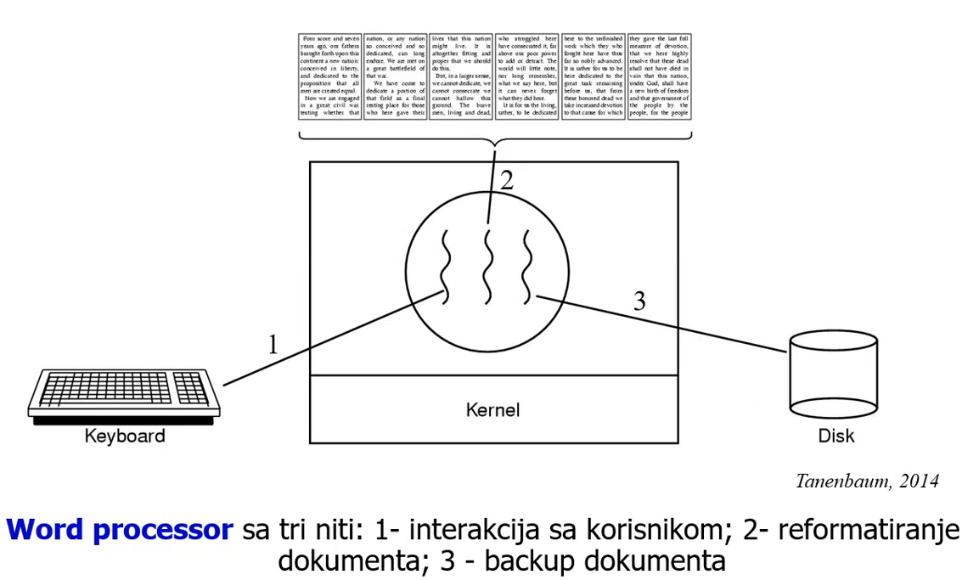
**Primer: RPC korišćenjem više niti**

* Program vrši dva poziva udaljenih procedura (PRC) na dva različita računara, i za svaki poziv koristi posebnu nit.
* Ostvareno je poboljšanje u performansama.

**Primer: Višenitnost na jednoprocesorskom računaru**

**Izvršavanje i sinhronizacija niti**

* U jednoprocesorskom sistemu niti jednog procesa se izvršavaju konkurentno, slično procesima kod multiprogramiranja.
* U višeprocesorskom sistemu niti jednog procesa se mogu izvršavati paralelno – na različitim procesorima.
* Niti jednog procesa nisu potpuno nezavisne, jer se izvršavaju u jednom adresnom prostoru, što znači da dele globalne promenljive i otvorene datoteke.
  1. Svaka promena memorisjkog resursa jedne niti utiče na druge niti koje koriste iste resurse.
  2. Neophodno je obezbediti sinhronizaciju više niti u pristupu istim memorijskim resursima.
  3. Tehnike za sinhronizaciju niti su identične tehnikama za sinhronizaciju procesa.

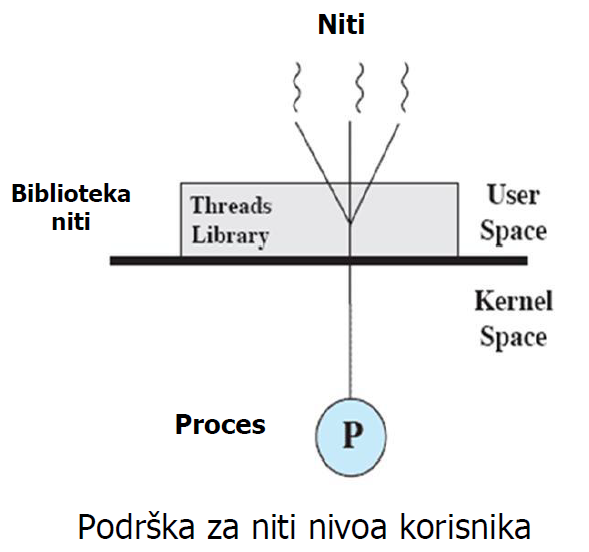
**Primer: Višenitni Word procesor i Web Server**

**Tipovi niti**

* Postoje dve kategorije implementacije niti:
  1. **Niti nivoa korisnika (ULT – User Level Thread)**
  2. **Niti nivoa jezgra (KLT – Kernel Level Thread)**
     + Kernelom podržane niti, lightweight procesi

**Niti nivoa korisnika – ULT**

* Kernel OS nije svestan postojanja niti.
* Nitima upravlja aplikacioni program koristeći biblioteku niti – skup funkcija za upravljanje nitima.
* Promena (komutiranje, switching) niti ne zahteva privilegije kernel moda, tj. Nema promene moda.
* Raspoređivanje niti je specifično za aplikaciju i obavlja se u okviru lokalne procedure u biblioteci niti, dakle bez promene moda (korisnički -> kernel).
* Svaki proces može imati sopstveni algoritam planiranja niti.
* Korišćenjem biblioteke niti bilo koja aplikacija se može implementirati kao višenitna.

**Niti nivoa korisnika – ULT**

Biblioteka niti sadrži kod za:

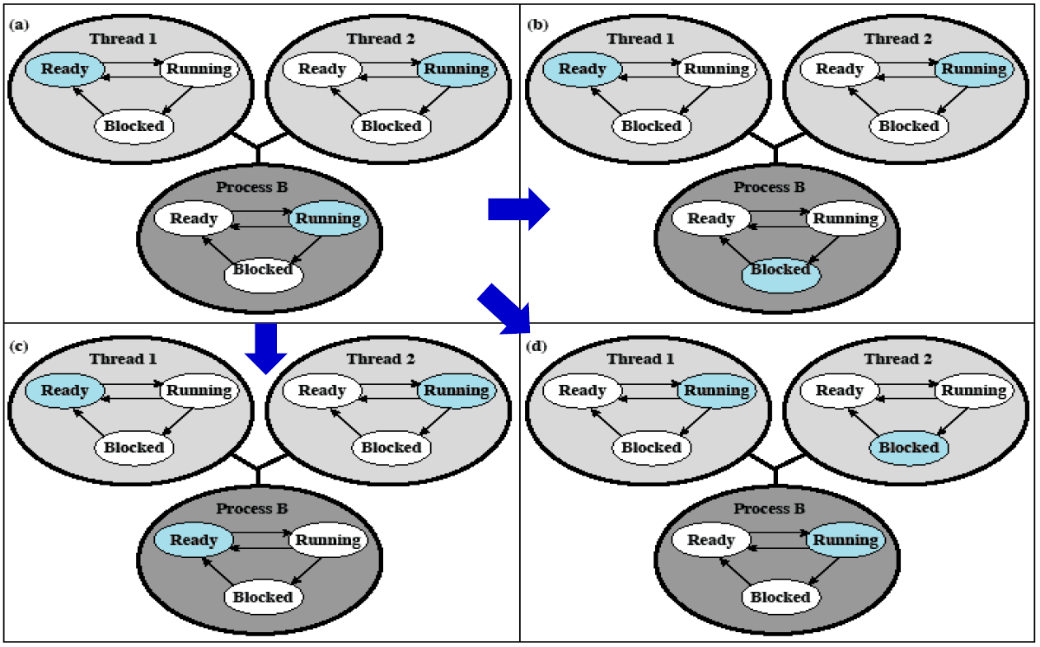
* Kreiranje niti
* Brisanje niti
* Prenos poruka i podataka između niti
* Raspoređivanje izvršenja niti
* Pamćenje i restauriranje konteksta niti...

Kontrolni blok niti (TCB) sadrži:

* Programski brojač
* Pokazivač magacina
* Registre procesora
* Stanje
* Prioritet...

**Niti nivoa korisnika – aktivnost kernela**

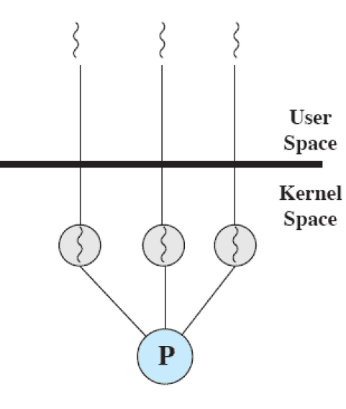
* Kernel ne vodi računa o aktivnostima niti, ali još uvek upravlja aktivnostima procesa
* Kada nit pozove neki sistemski poziv ( pozove funkciju OS-a), ceo proces se blokira, ali za biblioteku niti ta nit je još uvek u stanju izvršenja
* Stanja niti su nezavisna od stanja procesa.

**Stanja ULT niti i procesa**

1. Proces B je u stanju IZVRŠENJE i izvršava se NIT2
2. Kada se nit blokira, blokira se proces, dok je za biblioteku niti NIT2 još uvek u stanju IZVRŠENJE.
3. Ukoliko istekne vremenski kvant proces je SPREMAN, a NIT2 i dalje u stanju IZVRŠENJE
4. NIT2 se blokira kada zahteva neku akciju od NITI1, koja prelazi u stanje IZVRŠAVANJE, dok je proces sve vreme u stanju IZVRŠAVANJE.

**ULT – Prednosti i nedostaci**

* Nedostaci:
  + Mnogi sistemski pozivi su blokirajući i kernel blokira procese. Ukoliko jedna nit pozove blokirajući sistemski poziv, sve niti unutar tog procesa će biti blokirane:
    - Rešenje: korišćenje tehnike omotavanja (jacketing)
  + Kernel može dodeljivati procesore jedino procesima
    - Niti istog procesa se ne mogu istovremeno izvršavati na različitim procesorima.
* Prednosti:
  + Promena sa jedne na drugu nit ne zahteva privilegije kernel moda, nije potrebno prebacivanje moda (korisniči – kernel, kernel – korisnički), pa se brže odvija
  + Raspoređivanje može biti specifičan za proces; bira se najbolji algoritam.
  + ULT se može izdvojiti nad bilo kojim OS-om. Jedino je potrebna biblioteka niti koja se može implementirati u OS-u koji ne podržava niti.

**Niti nivoa kernela – KLT**

* Kernel upravlja nitima
* Nema biblioteke niti, ali postoji API za rad sa nitima
* Kernel održava konteksne informacije za procese i niti
* Kernel vrši planiranje i raspoređivanje niti
* Kernel vrši promenu niti
* Primeri : Windows, Linux, Mac OS

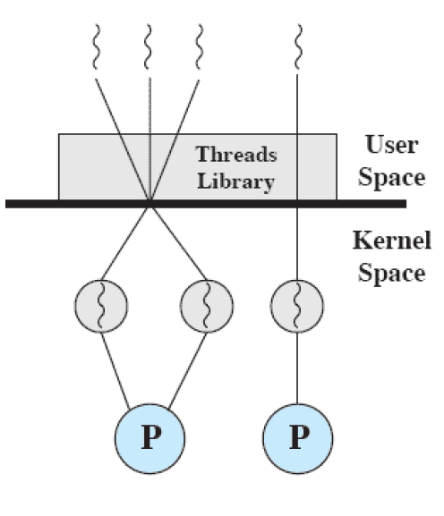
**KLT – Prednosti i nedostaci**

* Prednosti:
  + Kernel može simultano planirati više niti istog procesa na više procesora
  + Blokiranje se vrši na nivou niti – kada se nit blokira, jezgro bira spremnu nit istog ili drugog procesa
  + Programi kernela mogu biti višenitni
* Nedostaci:
  + Promena niti unutar istog procesa zahteva promenu moda (kernel mod) i angažovanje OS,
  + Upravljanje nitima nivoa kernela sporije je nego kod ULT-a.

**ULT vs. KLT vs. Procesi**

* Trajanje operacije nad nitima (ULT i KLT) i procesima
  1. Kreiranje prazne niti/procesa **(Null Fork)**
  2. Sinhronizacija niti/procesa zasnovana na signalima **(Signal Wait)**

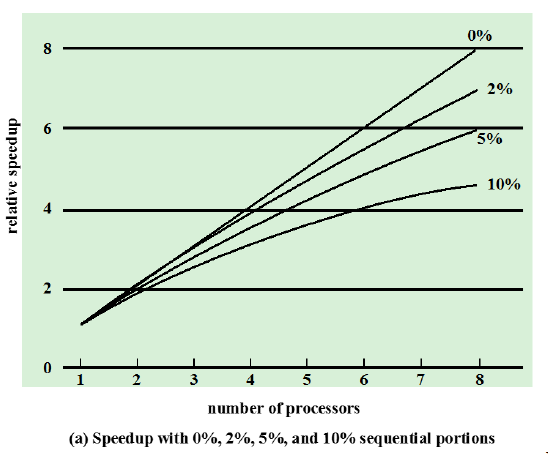
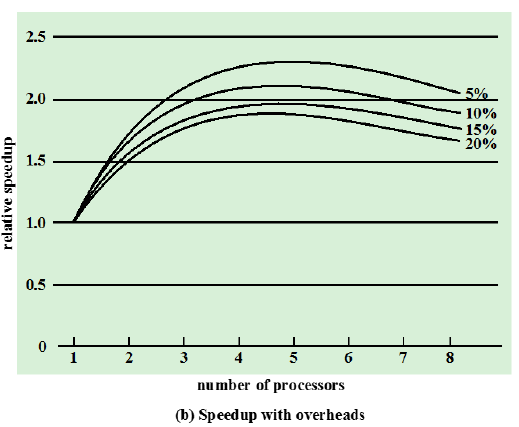
|  |  |  |  |
| --- | --- | --- | --- |
| **Operacija** | **ULT** | **KLT** | **Proces** |
| **Null Fork** | 34 | 948 | 11.300 |
| **Signal Wait** | 37 | 441 | 1.840 |



**Hibridna implementacija**

* Kreiranje niti obavlja se u prostoru korisnika
* Rapoređivanje i sinhronizacija niti se obavlja uglavnom na nivou aplikacije u prostoru korisnika
* Više ULT-a iz jednog procesa preslikavaju se u manji, ili jednak broj KLT-ova
* Programer može podešavati broj KLT-a za određenu aplikaciju
* Kombinuje dobre osobine ULT-a i KLT-a i minimizuje nedostatke
* Primer: Solaris

**Performanse softvera na više jezgara**

* Amdahl-ov zakon
  1. – procenat koda koji se može paralelizovati na N procesora
     + Overhead – dešava se prebacivanje procesa, menjanje registara, uključivanje OS-a itd.

**Niti u Windows-u, Solaris-u, Unix-u, Linux-u**

* Navodno ne treba