Niti, SMP i mikrokerneli

Operativni sistemi 1

Institut za matematiku i informatiku
Prirodno-matematički fakultet, Kragujevac

doc. dr Miloš Ivanović

Oktobar 2012. god.

O čemu će biti reči?

Koncept niti

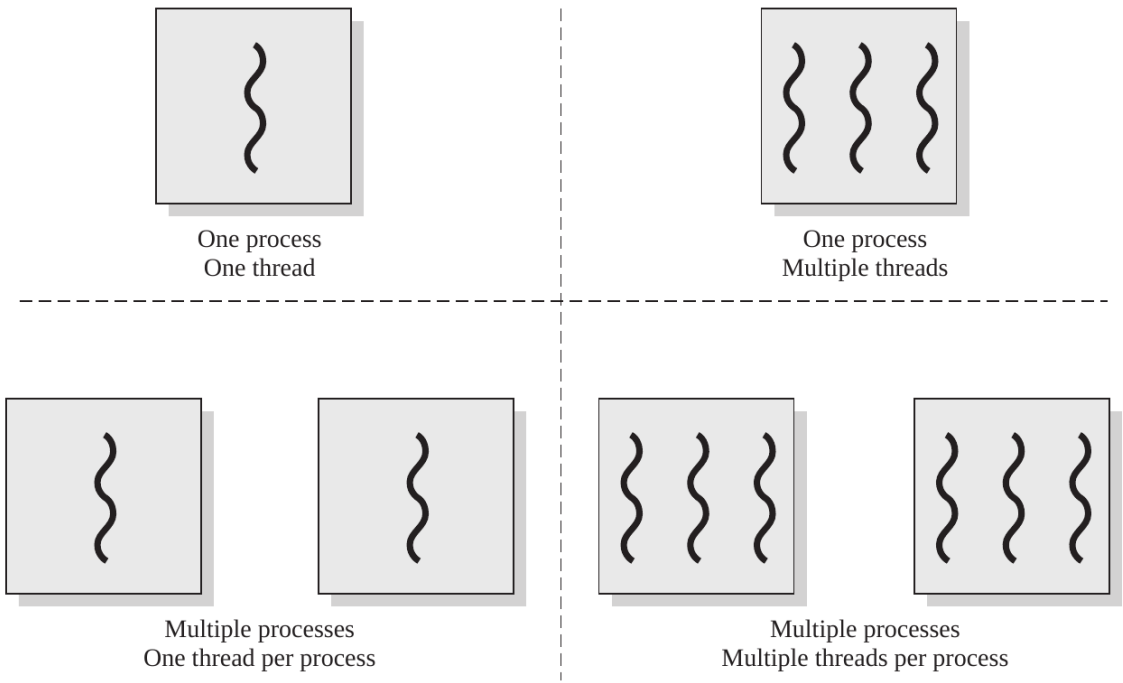
Definicija procesa koji je do sada prezentovan, obuhvata dva odvojena koncepta, i to:

1. **Vlasništvo nad resursima** - U datom vremenskom trenutku, procesu može biti dodeljena odgovarajuća memorija, stek, U/I uređaji, itd.
2. **Raspoređivanje/izvršavanje** - Izvršavanje procesa prati trag (*trace*) kroz jedan ili više programa. Njegovo izvršenje može da se prepliće sa izvršenjem ostalih procesa u sistemu.

Nit (*thread*)

Ukoliko se kocept izvršenja u potpunosti odvoji od procesa vlasništva nad resursima, onda je proces u mogućnosti da poseduje jedan ili više konkurentnih tragova izvršenja.

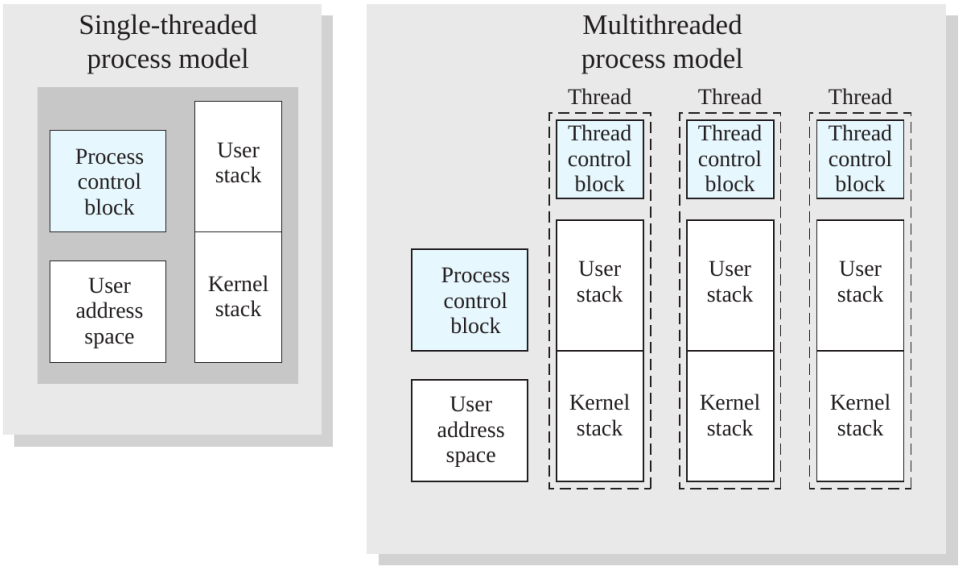
Multithreading



Vlasništvo procesa

Proces je i dalje vlasnik virtuelnog adresnog prostora, UI kanala, otvorenih fajlova isl.

Nadležnosti niti



1. **Stanje izvršenja niti** (*Running*, *Ready*, itd.)
2. Sačuvan **kontekst niti** kada se ne izvršava (PC, registri, ...)
3. **Zaseban stek**
4. **Prostor za lokalne varijable**
5. **Pristup memoriji svog procesa**, zajedno sa svim ostalim nitima tog procesa

Osnovne prednosti niti

1. Daleko je **kraće vreme potrebno za kreiranje niti** unutar procesa nego potpuno novog procesa
2. **Kraće vreme je potrebno i za terminaciju niti** u odnosu na vreme za terminaciju procesa
3. **Brže komutiranje niti u okviru istog procesa** u odnosu na komutiranje dva procesa
4. **Komunikacija između niti može da se obavi bez poziva kernela**, usled postojanja zajedničkog adresnog prostora

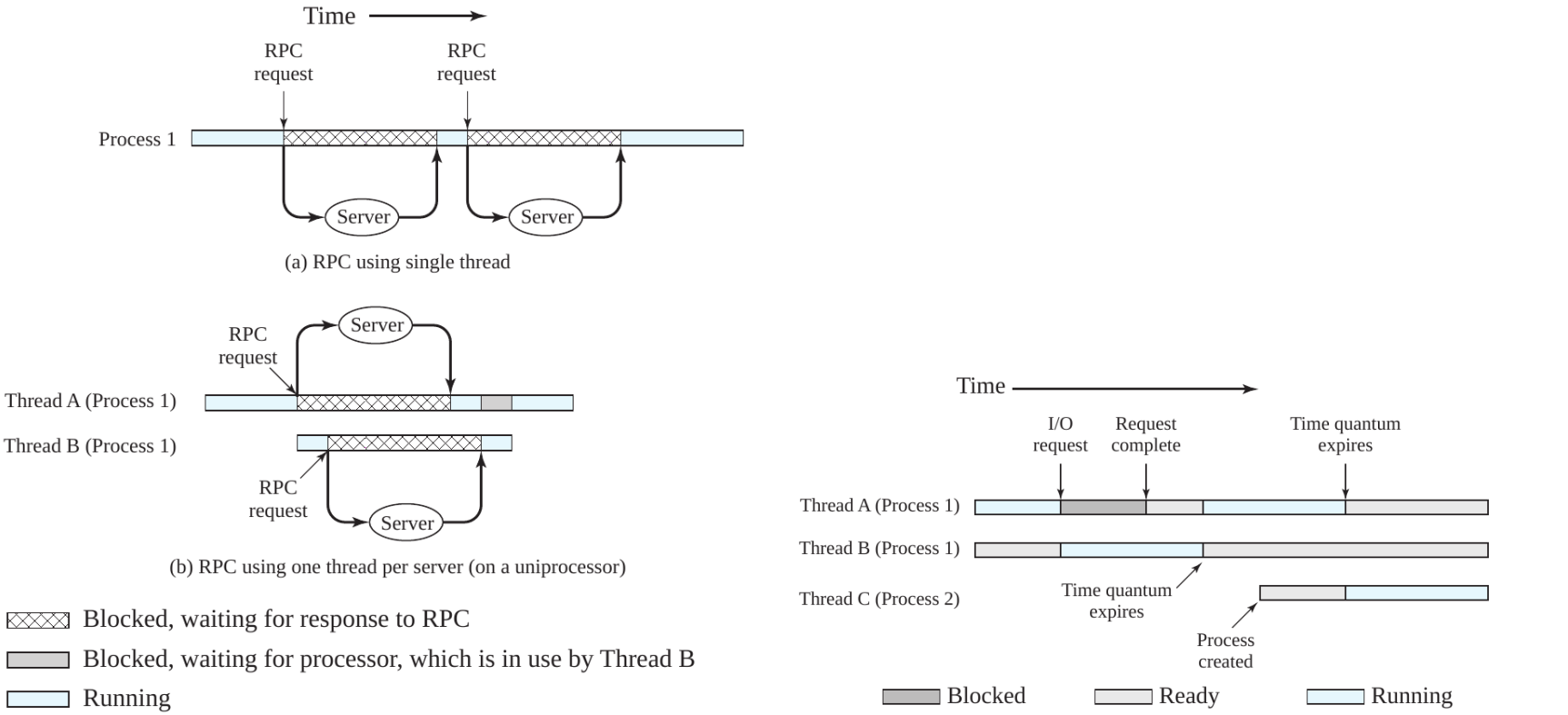
Primeri upotrebe niti

- **Pozadinske niti u GUI programima** - postiže se utisak o bržem odzivu sistema
- **Asinhrono procesiranje** - primer je nit koja u kancelarijskim paketima automatski u određenim intervalima snima *backup* fajla koji se obrađuje
- **Povećanje brzine izvršenja** - recimo jedna nit može da čita podatke sa diska i da često ulazi u blokirano stanje, dok druga može da obrađuje podatke. Na multiprocesorskim sistemima, ovo posebno dolazi do izražaja.
- **Modularna struktura programa** - Različite vrste aktivnosti mogu se implementirati putem niti.

Tipične operacije promene stanja niti

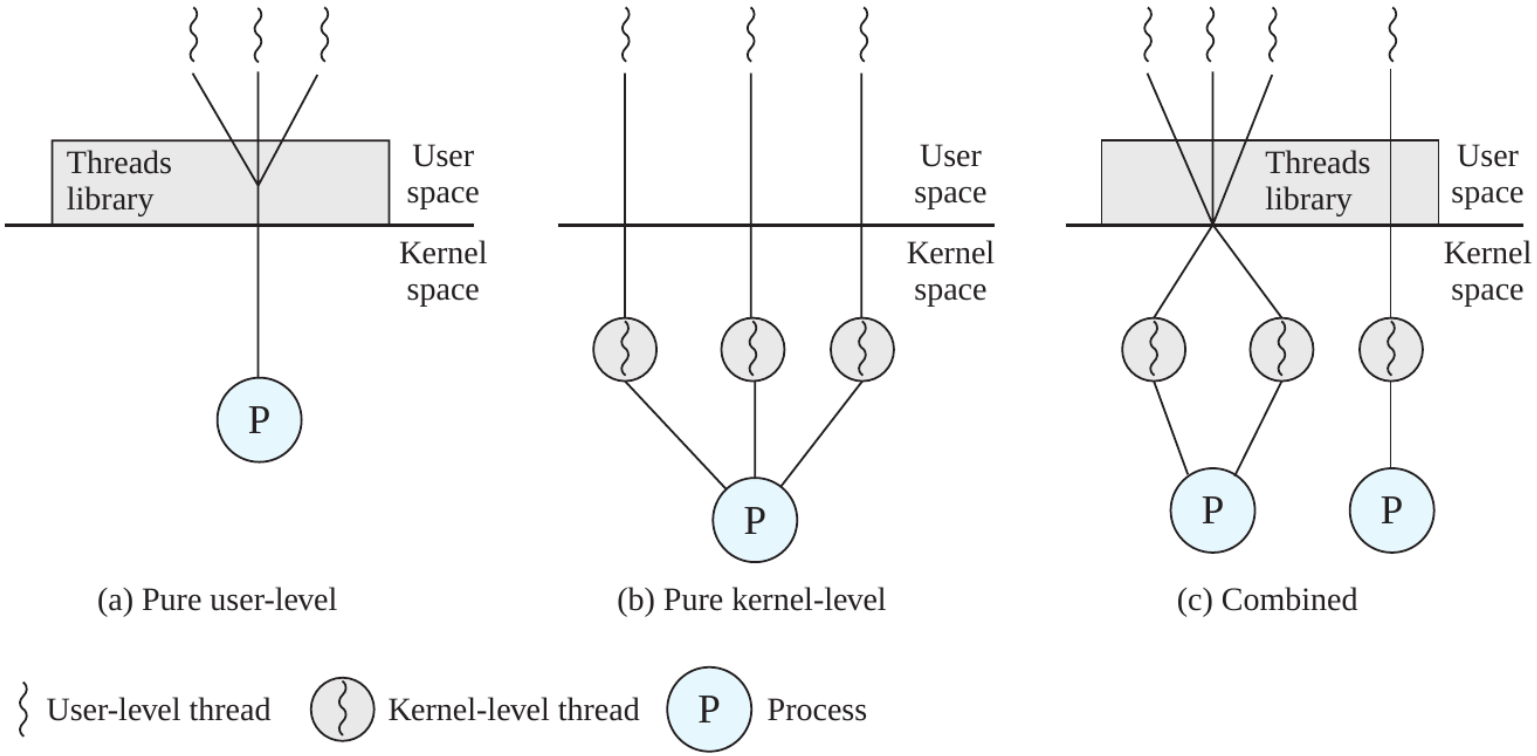
Spawn (kreiranje) niti, **Block** (nit čeka na događaj, čuva stanje, a procesor prelazi na obradu druge niti), **Unblock** (nit prelazi u *Ready* red), **Finish** (kontekst registara i stekova se dealocira).

Primeri upotrebe niti



Slika : **Levo**: Primer asinhronog poziva udaljene procedure
Desno: *Multithreading* u jednoprocesorskom okruženju

Niti korisničkog i kernel nivoa



Niti korisničkog nivoa (*User Level Threads*-ULT)

Prednosti

1. Prebacivanje niti ne zahteva privilegije kernela
2. Raspoređivanje niti može se uvesti specifično za datu aplikaciju
3. ULT-ovi se mogu izvršavati na svakom OS-u

Mane

1. Ako se blokira jedna nit, blokirane su sve niti unutar datog procesa
2. Ne mogu se iskorisitити prednosti multiprocesiranja

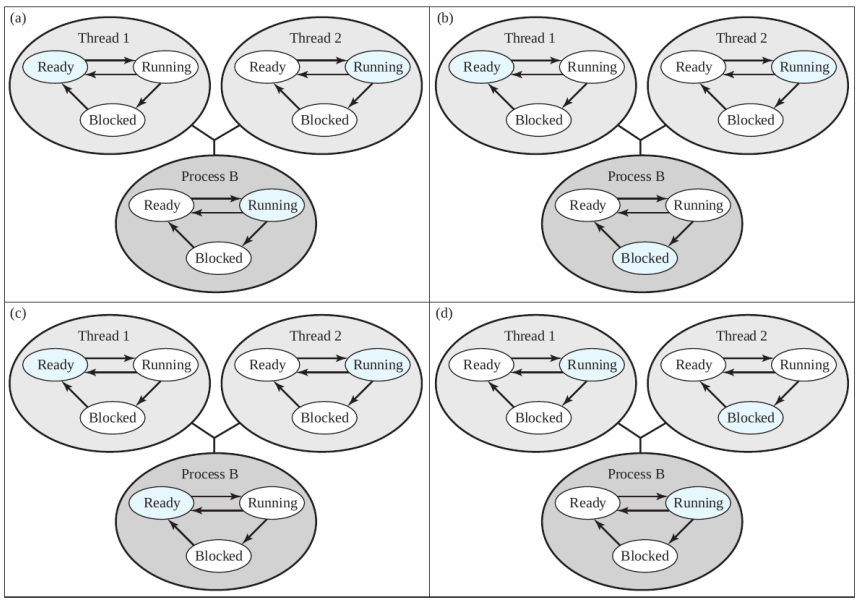
Oba problema se mogu rešiti "omotavanjem" blokirajućeg U/I poziva u neblokirajući.

Niti korisničkog nivoa (*User Level Threads*-ULT)

Primer

Sled koraka

1. Proces B izvršava nit 2 koja se blokira čekajući na U/I
2. Kontrola se predaje kernelu koji pokreće UI akciju, i daje kontrolu drugoom procesu
3. Kada proces B potroši svoj vremenski isečak, postaje *Ready*
4. Dolazi se do tačke kada biblioteka niti predaje kontrolul Niti 1 procesa B, pri čemu Nit 2 ostaje u stanju *Blocked*



Niti kernel nivoa (*Kernel Level Threads*-KLT)

Prednosti

1. Bilo koja aplikacija se može kreirati kao višenitna (primer je *Windows*)
2. Kernel može istovremeno rasporediti više niti jednog procesa na više procesora
3. Blokiranje jedne niti ne znači blokiranje procesa

Mane

1. Zahteva se prebacivanje u režim kernela i nazad za komutiranje niti

Oba problema se mogu rešiti "omotavanjem" blokirajućeg U/I poziva u neblokirajući.

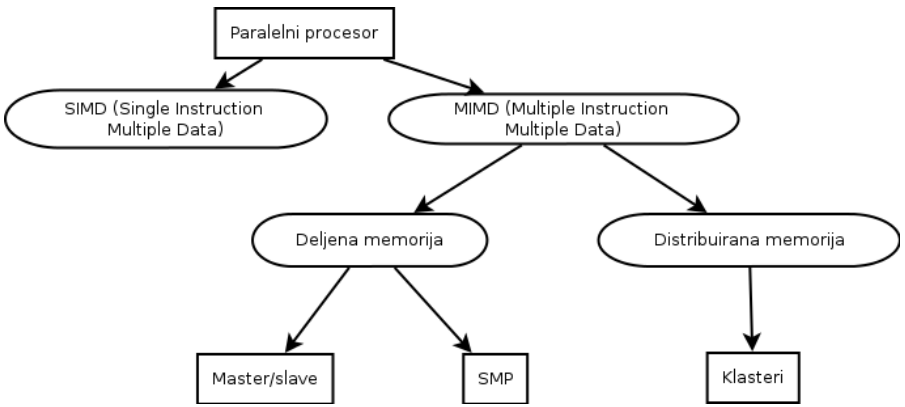
Symmetric MultiProcessing (SMP)

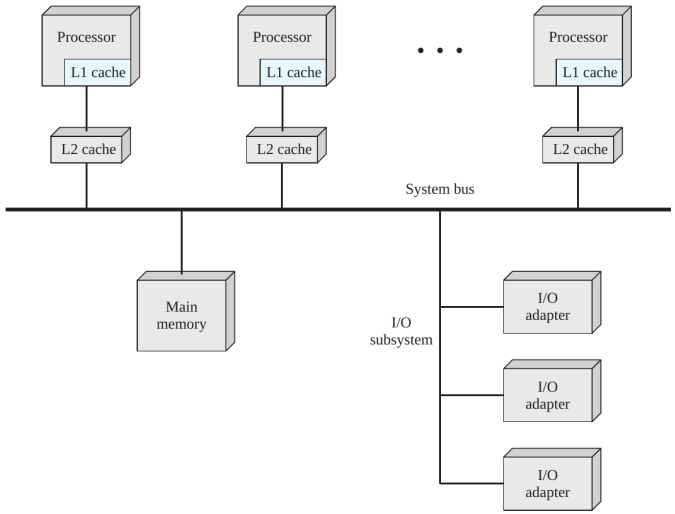
- Prva tehnologija paralelizacije obrade koja se pojavila još na Pentium procesorima je tzv. **pipeline**
- Već dugo su prisutni i SMP sistemi i klasteri (multikompjuteri)
- SMP se još naziva i **multiprocesor sa deljenom memorijom**
- Procesori preko deljene memorije komuniciraju

Flynn-ova taksonomija

1. Single Instruction Single Data (SISD)
2. Single Instruction Multiple Data (SIMD)
3. Multiple Instructions Single Data (MISD)
4. Multiple Instructions Multiple Data (MIMD)

Arhitekture paralelnih procesora





Keš memorija kod SMP-a

Na savremenim procesorima postoji bar jedan nivo keš memorije kao privatni keš procesora. Može se javiti **problem koherentnosti keša**, koji se obično rešava hardverski.

Pitanja projektovanja OS-a koji radi na SMP-u

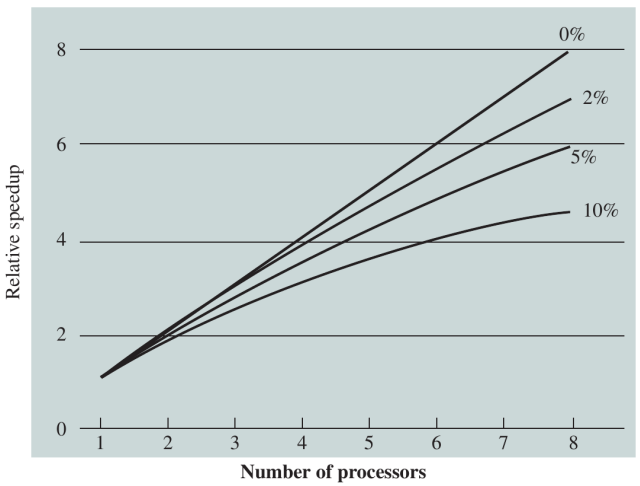
Konkurentni procesi/niti, raspoređivanje, sinhronizacija, upravljanje memorijom, pouzdanost, postepeni otkaz.

Performanse Multicore arhitekture

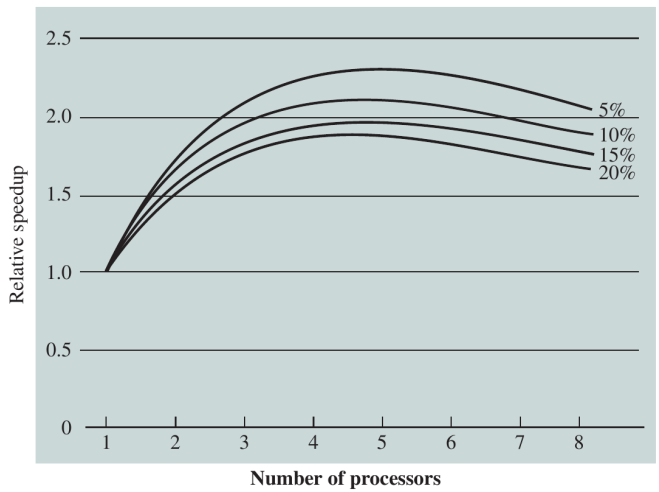
- $\sigma(n)$ - vreme potrebno da se izvrši inherentno sekvencijalni deo
- $\phi(n)$ - vreme potrebno da se izvrši deo koji se može paralelizovati
- $\kappa(n, p)$ - dodatak (*overhead*)

Amdalov zakon

$$Speedup(n,p) = \frac{\sigma(n)+\phi(n)}{\sigma(n)+\frac{\phi(n)}{p}+\kappa(n,p)} \rightarrow Speedup \leq \frac{1}{(1-f)+\frac{f}{p}}$$

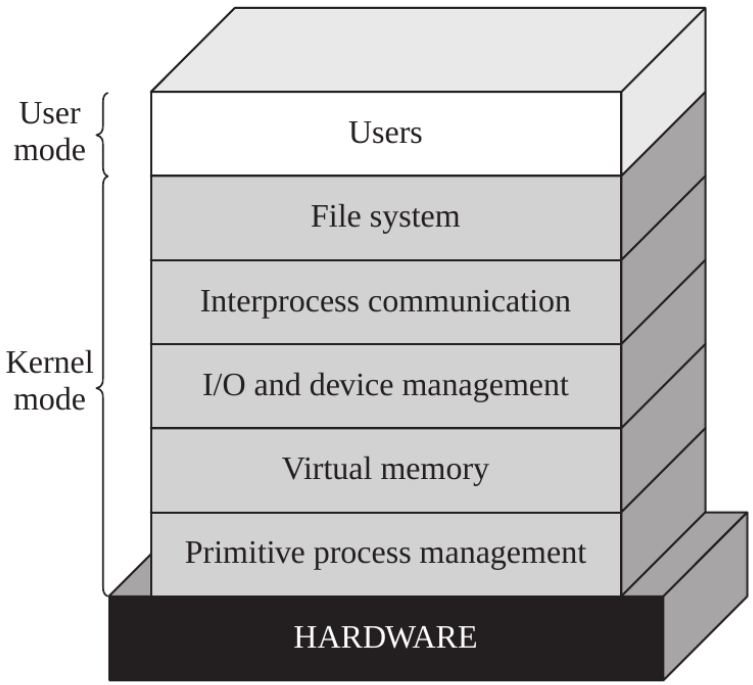


(a) Speedup with 0%, 2%, 5%, and 10% sequential portions

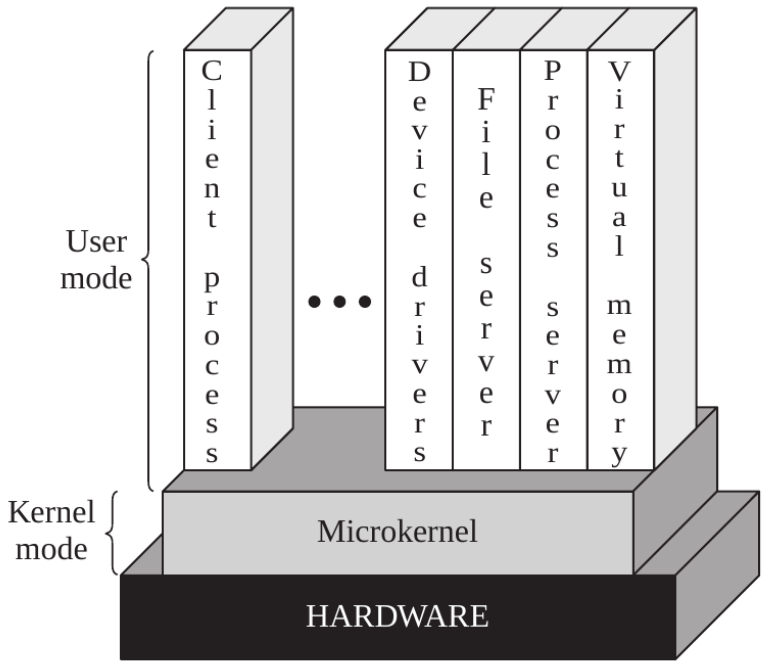


(b) Speedup with overheads

Mikrokernel



(a) Layered kernel



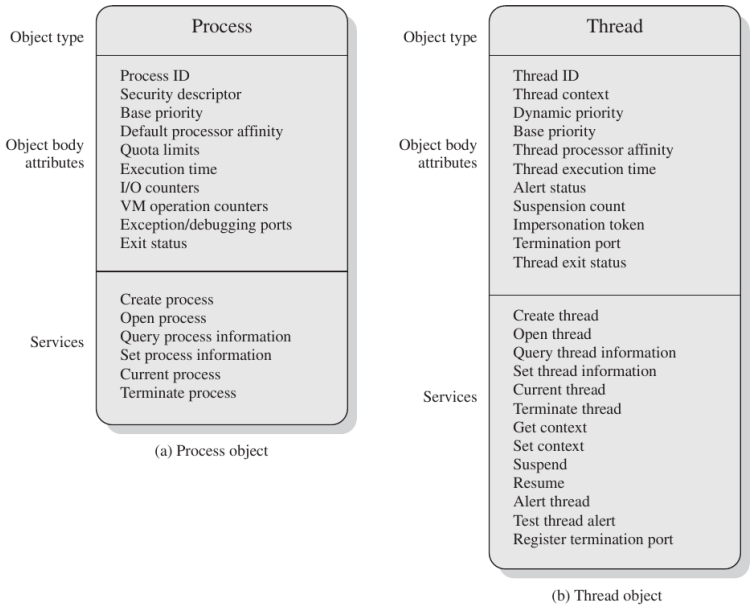
(b) Microkernel

Prednosti mikrokernela

1. Uniformni interfejsi (nema razlike između zahteva procesa korisničkog i kernel nivoa)
2. Proširivost
3. Fleksibilnost
4. Prenosivost
5. Pouzdanost
6. Podrška distribuiranim sistemima
7. Podrška za objektno orijentisane tehnologije

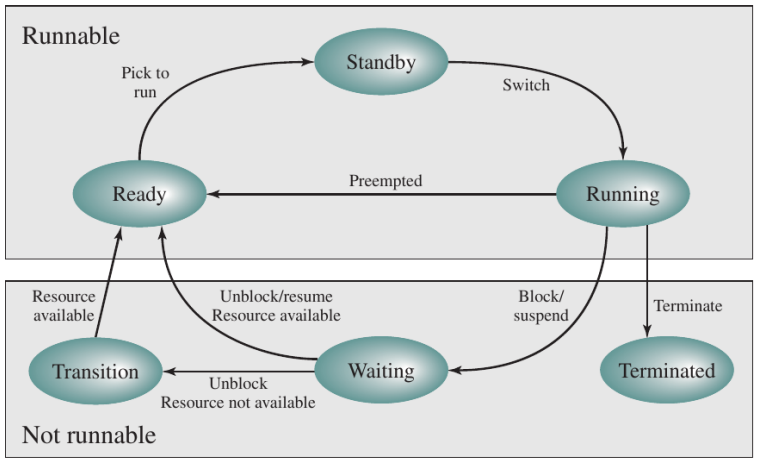
Procesi i niti kod Windowsa

- Windows procesi su implementirani ako objekti
- I procesi i niti imaju ugrađene mehanizme za sinhronizaciju
-



(a) Process object

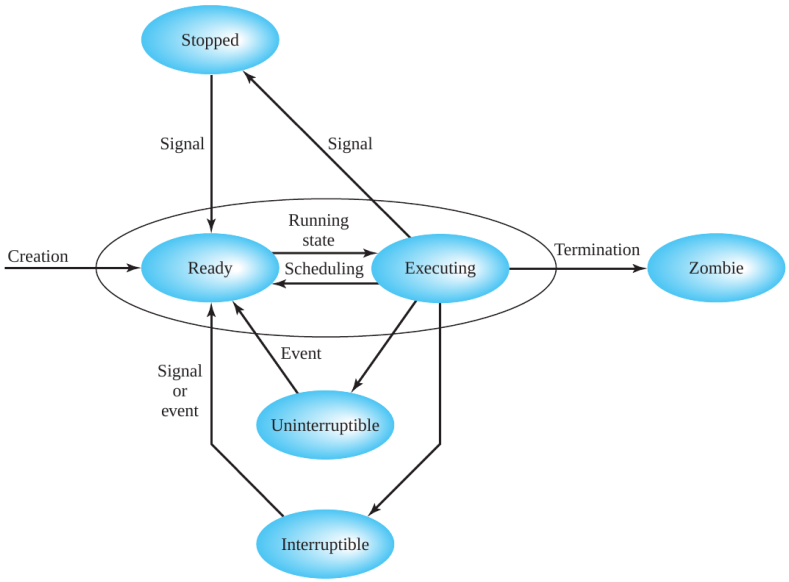
(b) Thread object



Procesi i niti kod Linuxa

Struktura `task_struct`

1. Stanje
2. Informacije o raspoređivanju
3. Identifikatori
4. Međuprocesna komunikacija
5. Linkovi
6. Vremenski brojači
7. Fajl sistem
8. Adresni prostor
9. Kontekst procesora



Procesi i niti kod Linuxa

- POSIX threads je biblioteka niti korisničkog nivoa
- Kako se ove niti preslikavaju u niti kernel nivoa
- Na Linuxu se nit kreira **kloniranjem** PCB-a niti roditelja
- Na taj način niti istog procesa dele sve strukture
- Svaka ovako kreirana nit ima sopstveni stek