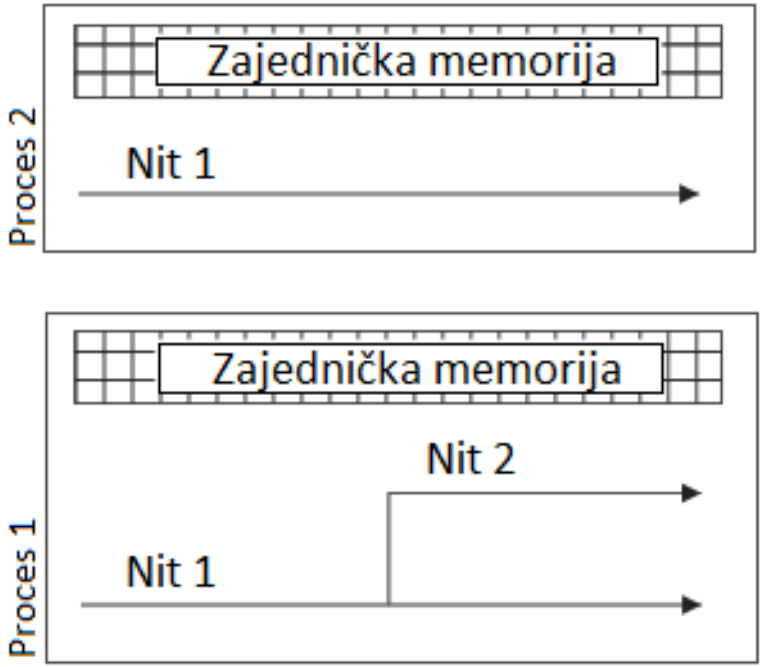
Konkurentno i paralelno programiranje

IKP

Pojam programske niti (thread)

- Aktivnost procesa karakteriše redosled u kome se izvršavaju naredbe programa. Ako je taj redosled određen u vreme programiranja, onda može da se prikaže kao **nit** koja povezuje izvršavanje naredbi u redosledu njihovog izvršavanja
- Niti predstavljaju osnovnu jedinicu operativnog sistema koje konkurišu za vreme procesora
- Svaki proces se pokreće izvršavanjem jedne niti - main
- U toku izvršavanja procesa, moguće je stvoriti proizvoljan broj niti iz bilo koje predhodno formirane niti

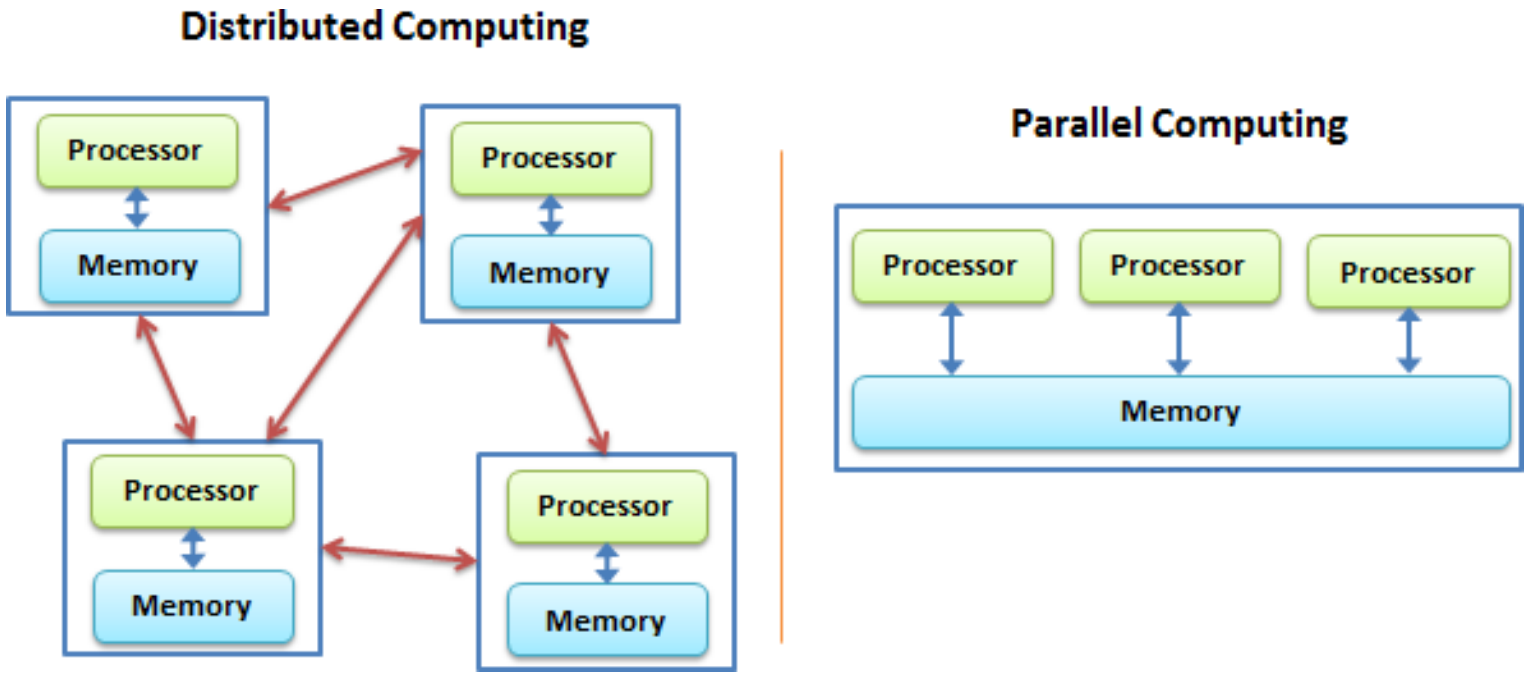


Konkurentno programiranje

- Paralelno != Konkurentno
- Jednoprocesorski računar
- Procesi sa više istovremeno (*concurrently*) postojećih niti se nazivaju konkurentni procesi
- Konkurišu za procesorsko vreme
- Istovremenost izvršavanja programskih niti je prividna
- Procesor izvršava naredbe jedne niti dok je moguća njena aktivnost ili dok se ne desi prekid. Ako obrada prekida izazove preključivanje, u nastavku svog rada procesor izvrši naredbe potprograma preključivanja i zatim produži sa izvršavanjem naredbi druge niti
- Preplitanje programskih niti

Paralelno vs. Distribuirano programiranje

- Paralelno programiranje – Jedan računar, više procesora (ili jezgara), deljena memorija
- Distribuirano programiranje – Više računara



Sprečavanje štetnih preplitanja

- Međusobna isključivost (*mutual exclusion - mutex*)
  - Rešava problem štetnog preplitanja (druga nit pristupa deljenoj promenljivoj pre nego što je prva nit završila operaciju nad njom – promenljiva moze biti u nekonzistentnom stanju)
- Kritične sekcije
  - Tela operacija deljenih klasa ili delovi ovih tela, čije izvršavanje je kritično za konzistentnost deljenih promenljivih, se nazivaju kritične sekcije
- Sinhronizacija
  - Međusobna isključivost kritičnih sekcija se ostvaruje vremenskim usklađivanjem njihovih izvršavanja - sinhronizacija
- Atomski regioni
  - U toku izvršavanja ovih kritičnih sekcija nisu moguće ni obrade prekida – zbog toga treba da budu što kraći
- Propusnice

Kreiranje niti

- Kreiranje niti je vremenski i memorijski zahtevna operacija
- Nakon završetka rada niti, potrebno je osloboditi handle na nju

```
HANDLE CreateThread(LPSECURITY_ATTRIBUTES lpThreadAttr, DWORD dwStackSize,
          LPTHREAD_START_ROUTINE lpStartAddress, LPVOID lpParameter,
          DWORD dwCreationFlags, LPDWORD lpThreadId);

BOOL CloseHandle(HANDLE hObject);
```

Thread pool

- Prethodno pomenuti problem kreiranja niti (vremenski skupa operacija) se rešava upotrebom **thread pool**-a
- Prilikom inicijalizacije thread pool-a se kreira određen broj niti u zavisnosti od veličine thread poola
- Umesto kreiranja nove niti, uzima se već spremna nit iz thread poola, te se ne troši vreme na kreiranje niti
- Nakon završetka zadatka koji je nit izvršila, nit se vraća u thread pool.
- Kako rešiti problem kada se potroše sve niti iz poola?

Kritična sekcija

- Kritična sekcija obezbenuje mehanizam nedeljivog pristupa deljenim resursima

- **VOID InitializeCriticalSection(LPCRITICAL_SECTION** *lpCriticalSection*);
- **VOID EnterCriticalSection(LPCRITICAL_SECTION** *lpCriticalSection*);
- **VOID LeaveCriticalSection(LPCRITICAL_SECTION** *lpCriticalSection*);
- **VOID DeleteCriticalSection(LPCRITICAL_SECTION** *lpCriticalSection*);

Semafori

- Koriste se za sinhronizaciju izvršavanja niti u okviru jednog procesa (ili veše na istom računaru).

- **HANDLE CreateSemaphore(LPSECURITY_ATTRIBUTES** *lpSemAttr*, **LONG** *lInitialCount*, **LONG** *lMaximumCount*, **LPCTSTR** *lpName*);

- **DWORD WaitForSingleObject(HANDLE** *hHandle*, **DWORD** *dwMilliseconds*);

- **BOOL ReleaseSemaphore(HANDLE** *hSemaphore*, **LONG** *lReleaseCount*, **LPLONG** *lpPreviousCount*);

- **BOOL CloseHandle(HANDLE** *hSemaphore*);

Dead-lock

- Dve (ili više) niti čekaju jedna na drugu da bi nastavile sa radom
- Obično se dešava prilikom nepravilnog korišćenja kritičnih sekcija i semafora
- Broj zauzimanja kritične sekcije označen sa N, a broj oslobananja sa M.

Situacija	Opis
N = M	Kritična sekcija pravilno oslobonena
N > M	Kritična sekcija i dalje zauzeta
N < M	Oslobananje nezauzete kritične sekcije od strane jedne niti može izazvati beskonačno čekanje druge niti pri pokušaju zauzimanja iste kritične sekcije

- Kada se nit završi, kritična sekcija mora da bude slobodna. U protivnom može doći do zaustavljanja rada ostalih niti ako one čekaju na oslobananje te kritične sekcije (eng. deadlock).

Lock - contention

- Veliki broj niti pokušava da pistupi istom deljenom resursu
- Ako više niti čita istu deljenu promenljivu, dok druge niti menjaju tu deljenu promenljivu, poželjno je koristiti **Slim reader/writer lock**
- Više niti u isto vreme može da čita, dok samo jedna nit može da menja deljeni resurs
- **VOID WINAPI InitializeSRWLock(_Out_ PSRWLOCK SRWLock);**
- **VOID WINAPI AcquireSRWLockShared(_Inout_ PSRWLOCK SRWLock);**
- **VOID WINAPI ReleaseSRWLockShared(_Inout_ PSRWLOCK SRWLock);**
- **VOID WINAPI AcquireSRWLockExclusive(_Inout_ PSRWLOCK SRWLock);**
- **VOID WINAPI ReleaseSRWLockExclusive(_Inout_ PSRWLOCK SRWLock);**

Context switch

- Context switch predstavlja proces promene aktivne niti na procesoru
- Taj proces obuhvata snimanje stanja trenutne niti (registre, stack), te čitanje snimljenog stanja sledeće spremne niti i njen nastavak rada
- Meri se brojem context switch-eva na procesoru u jednoj sekundi
- Ako je veliki broj context switch-eva, to signalizira na postojanje nekog problema