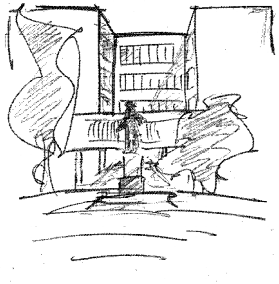
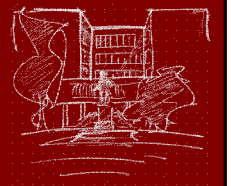
# Развој софтвера 11а



Саша Малков
Универзитет у Београду
Математички факултет
2023/2024

[P290]
## Развој софтвера
Саша Малков



Тема 14

# Конкурентно програмирање (наставак)





## Функције и више нити

- Функције (и други потпрограми) могу имати релативно сложен однос са конкурентним окружењем
- Упштено, функције могу бити:
  - са јединственим позивањем
  - са поновљеним позивањем (енгл. *reentrant*)
  - безбедне по нити (енгл. *thread-safe*)

Универзитет у Београду - Математички факултет





## Функције са јединственим позивањем

- Кажемо да је функција са јединственим позивањем ако не сме бити истовремено употребљавана у различитим нитима
- Могући узроци
  - у телу функције се користе глобални подаци или други глобални ресурси на начин који није безбедан
  - ти глобални ресурси се употребљавају независно (или бар не непосредно зависно) од наведених аргумената

Универзитет у Београду - Математички факултет

## Функције са поновљеним позивањем

- Кажемо да је функција са поновљеним позивањем ако сме да се истовремено употребљава у различитим нитима, али само уз претпоставку да се не употребљава на истим подацима
- Могући узроци
  - употреба података или ресурса на начин који није безбедан







## Функције безбедне по нити

- Кажемо да је функција безбедна по нити ако сме да се истовремено употребљава у различитим нитима без посебних ограничења
- Функција је безбедна по нити ако све податке (осим локалних аутоматских) употребљава уз пуно уважавање специфичности конкурентног окружења







## Класе и више нити

- Све што важи за функције важи и за класе
- Класа је
  - са јединственим позивањем
    - ако је бар један метод класе са јединственим позивањем
  - са поновљеним позивањем
    - ако ниједан метод није са јединственим позивањем,
    - бар један метод је са поновљеним позивањем и
    - ниједном податку не може да се непосредно приступа
  - безбедна по нити
    - ако су сви методи безбедни по нити и
    - ниједном податку не може ад се непосредно приступа



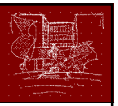




## Проблеми при писању конкурентних програма

- Превиђање проблема дељења података при писању функција и класа
- Превиђање неприлагођености употребљаваних класа (функција) конкурентним окружењима
  - тј. употребљавају се као да су безбедне по нити, а да оне то заправо нису

## Како писати код безбедан по нити?



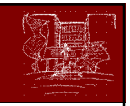
- Писање чисто функционалног кода
- Употреба локалних података и података који су локални за нит
- Обезбеђивање међусобног искључивања
- Обезбеђивање атомичних операција







## Како писати безбедан код?



- **Писање чисто функционалног кода**
  - потпрограми се пишу тако да
    - користе само аргументе и локалне промењиве
    - не мењају аргументе
  - потпрограми се позивају само са безбедним аргументима
    - тј. аргументима који не могу да буду мењани у другим нитима
    - аргументи преносе по вредности
    - ако се преносе по адреси, онда мора да буде сигурно да се неће мењати у другим нитима
- Употреба локалних података и података који су локални за нит
- Обезбеђивање међусобног искључивања
- Обезбеђивање атомичних операција







## Како писати код безбедан по нити?

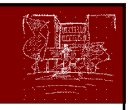
- Писање чисто функционалног кода
- **Употреба локалних података и података који су локални за нит**
  - Такве податке ниједна друга нит не може да користи
  - Самим тим, чим су такви подаци безбедни, безбедан је и потпрограм који само њих употребљава
- Обезбеђивање међусобног искључивања
- Обезбеђивање атомичних операција







## Како писати код безбедан по нити?



- Писање чисто функционалног кода
- Употреба локалних података и података који су локални за нит
- **Обезбеђивање међусобног искључивања**
  - мутекси, катанци,...
- Обезбеђивање атомичних операција

## Како писати код безбедан по нити?

- Писање чисто функционалног кода
- Употреба локалних података и података који су локални за нит
- Обезбеђивање међусобног искључивања
- **Обезбеђивање атомичних операција**
  - библиотеке за атомичне операције, критичне секције, мутекси, катанци,...







## Неки принципи

- Не одлагати старање о безбедности нити за касније
  - од самог почетка пројектовања неке класе или функције мора се имати у виду да ли се планира употреба у окружењу са више нити
  - накнадно прилагођавање може да буде веома скупо, па и уз скоро потпуно преправљање класа







## Неки принципи (2)

- Не стављати катанце и мутексе *одока*
  - катанци, мутекси и слични механизми за изоловање ће учинити код безбедним *само ако се добро употребљавају*
  - међутим, истовремено ће и спустити ниво паралелности и искоришћености вишепроцесорске машине
  - боље је стављати више мањих катанаца него мање већих
    - ситнији катанци смањују вероватноћу чекања нити
  - повећавање броја катанаца ствара услове за настајање мртвих петљи
  - ПАЖЉИВО!!!



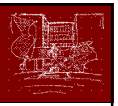




## Неки принципи (3)

- Не штитити само код или само податке
  - значајна унапређења квалитета програма и нивоа перформанси се могу постићи ако се закључава *права врста ресурса*
  - некада је боље штитити податке, али не увек
    - најчешће је добро штитити податке
    - ако је много података који се деле, долази до опасности од мртвих метљи
  - некада је боље штитити код, али не увек
    - обично је боље штитити код једним катанцем (мутексом,...) него податке са много катанаца (мутекса,...)
    - али ако се подаци које тај код чита/мења могу користити и на неком другом месту, онда то није довољно

## Неки принципи (4)



- Дељење података
  - избегавати дељење података међу нитима
  - посебно избегавати мењање истих података од стране више нити
  - ако је дељење неопходно, обавезно закључавати (податке или код)
  - минимизовати дељење података
- Закључавање
  - избегавати дуготрајна закључавања
  - ако је потребно поставити већи број катанаца, увек их постављати истим редом (макар и азбучним)
    - или користити технике “истовременог” закључавања







## Неки принципи (5)

- Модел конкурентности
  - строго дефинисати улогу сваке нити
  - ако је нит намењена тачно једном послу, не омогућавати јој приступ подацима и коду који се не односе на тај посао
  - стриктним везивањем нити за што ужи простор кода и података смањује се простор за сукобљавање нити око дељених ресурса
  - безбедније је направити више специјализованих него мање општих нити
  - документовати модел конкурентности
- Не старати се о безбедности по нити у коду за који то није потребно
  - то је губљење времена током развоја
  - постављање сувишних катанаца спушта перформансе







## C++ STL

- Стандардна библиотека је имплементирана са високим перформансама као једним од примарних циљева
- Није уграђено старање о конкурентности и дељењу ресурса
- Колекције се морају експлицитно закључавати при дељењу





### Пример имплементације критичних региона

```
static bool flags[2];      // označava koja nit želi da uđe u kr.region
static int turn = 0;       // označava koja nit je na redu


void EnterCriticalRegion(int i) // i će uvek biti 0 ili 1
{
   int j = 1 - i;          // indeks druge niti
   flags[i] = true;        // označimo želju za ulazak u kr.region

   while (flags[j]){       // čekamo sve dok je i druga nit zainteresovana
      if (turn == j){      // nije naš red - povlačimo se i čekamo dalje
         flags[i] = false;
         while (turn == j) /* "busy wait" */;
         flags[i] = true;
      }
   }
}


void LeaveCriticalRegion(int i)
{
   turn = 1 - i;           // prepuštamo red drugoj niti
   flags[i] = false;       // napuštamo region
}
```

## Проблем прет. решења

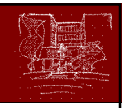
- Савремени процесори умеју да измене редослед извршавања инструкција које не раде на истим подацима
- Чекање у петљи

Универзитет у Београду - Математички факултет





## Међупроцесна комуникација



- Неки од уобичајених видова међупроцесне комуникације (енгл. *inter-process communication - IPC*) су:
  - сигнали
  - сокети (*sockets*)
  - слање порука
  - редови порука (*message queues*)
  - цеви (*pipes*)
  - именоване цеви (*named pipes*)
  - семафори,...
  - дељена меморија
  - датотеке пресликане у меморију (*memory mapped files*)
  - датотеке

Универзитет у Београду - Математички факултет





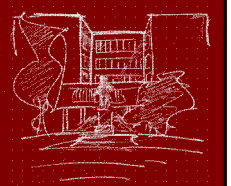
## Комуникација међу нитима

- Комуникација између процеса се може одвијати искључиво путем комуникационих канала обезбеђених на нивоу оперативног система или рачунарске мреже
- Комуникација између нити се може одвијати и путем ресурса који се деле међу нитима, као што су меморија или отворене датотеке
  - Мора се водити рачуна о исправном дељењу комуникационих ресурса, тј. о атомичности операција на дељеним ресурсима
  - Често је најбоље и за комуникацију међу нитима користити механизме намењене за процесе

Универзитет у Београду - Математички факултет



## Хвала на пажњи!



МАТФ
Универзитет у Београду
Математички факултет