

Programovanie v operačných systémoch

09 - Asynchronous programming

Jozef Šiška



Department of Applied Informatics
Comenius University in Bratislava

2016/2017

Asynchronous programming

- using asynchronous operations in a (usually) single threaded program
 - non-blocking IO and select / epoll
 - timers
 - events
- mainloop + event registration
- UI, highly performant IO servers

comparison to threads

Advantages

- simpler implementation when too many things interact (no locking, no need to think about what runs where)
- more efficient in highly asynchronous loads (server IO) - no kernel roundtrips for scheduling, synchronization

Disadvantages

- harder / more complicated to write without a good framework / library
- everything needs to be asynchronous - one blocking operation breaks everything (dns resolving?, big local files)
 - can be solved by "pushing" the operation to another thread, but can be complicated without good libraries>
- can't utilize multiple core efficiently by default, but good frameworks can...
 - ... and synchronization can still be avoided most of the time
- "inversion of control" - again can be helped with good framework or language design

Implementation

- cancellations, bookkeeping of "subscriptions"
- "callback hell" (older javascript, C), breaks code flow
 - lambdas, inline function syntax helps a bit, but not much
- listeners, observer pattern (OOP) - a lot of boilerplate, breaks code flow
- signals and slots
- futures, then-ables, continuations
- async frameworks - async / await in C# and javascript (and python?)
 - can be "emulated" in any language that can save and resume function execution, such as generator functions