



DelphiDay

italian conference

THREAD-SAFETY

How to write thread-safe code



PAOLO ROSSI

WINTECH ITALIA - CTO

SENCHA & EMB. MVP



 blog.paolorossi.net

 paolo@paolorossi.net

 twitter.com/awebguy

 github.com/paolo-rossi

 linkedin.com/in/paolo-rossi-pc



DelphiDay
italian conference

19-20 Giugno 2025
Piacenza



wintech
italia



GITHUB PROJECTS



github.com/paolo-rossi



Delphi JWT

JSON Web Token Library



WiRL

REST Library for Delphi



Linux Daemon

Real Linux daemons



Delphi Neon

JSON Serialization Library



OpenAPI-Delphi

OpenAPI 3.0 Library



NATS Delphi

NATS Client Library for Delphi



AGENDA

1. Introduction
2. Threads
3. Multi-thread concepts
4. Thread-safety: concepts
5. Synchronization
6. Legacy code



Introduction

1



OLD CODE NEW CHALLENGES

- You probably have a C/S application
 - Forms, DataModules
- You want to (have to) build some REST, SOAP, TCP, Server
- Of course you want to reuse your old code as much as possible
 - Can you?
 - Is it safe?
 - Better to rewrite from scratch?



IN THIS SEMINAR...

- It's not a general-purpose multi-thread course
- Focused on thread-safety
- Focused on old code refactoring
 - To be used together in new applications
 - To be used in hybrid applications

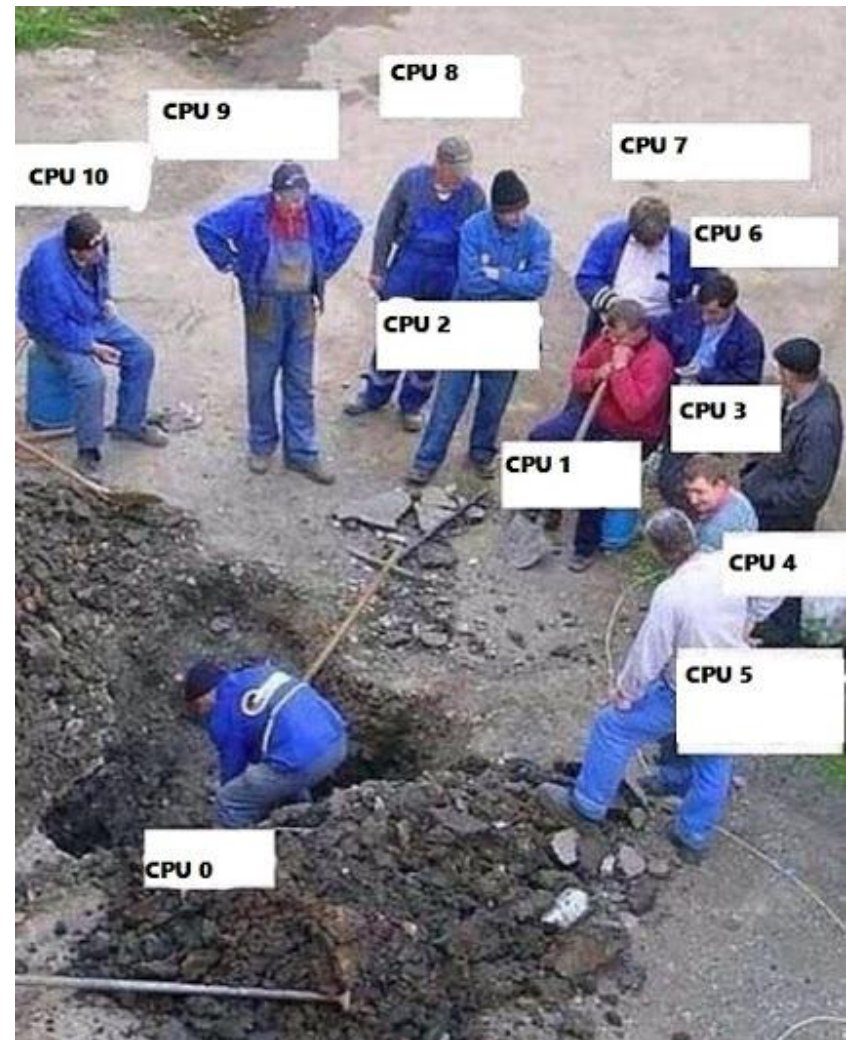


Threads

2



“Standard” Delphi Application in a Multicore System





BENEFITS

- Responsiveness
- Speed*
- Better resource utilization
- Better encapsulation of concepts
- Simpler Program Design*

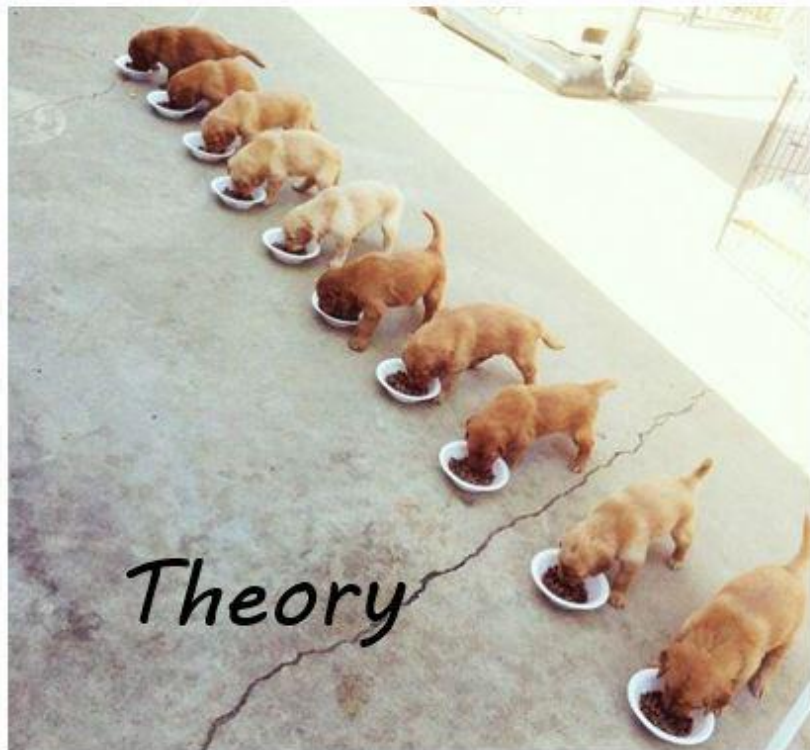


COSTS

- Context Switching Overhead
- Increased Resource Consumption
- Harder to write (correct) code
- Harder to debug and test
- Harder to port existing code



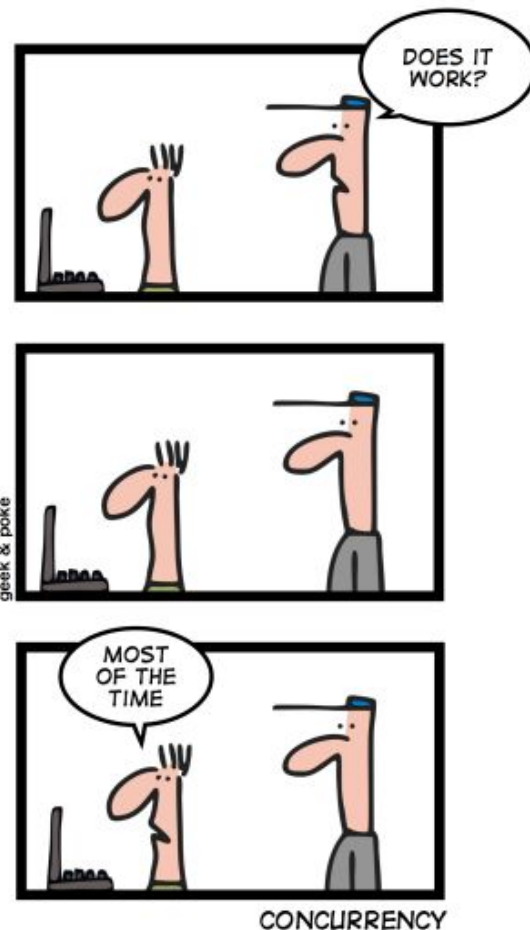
Multithreaded programming





Debugging a Multi-Thread Application

SIMPLY EXPLAINED





MT Concepts

3



CONCEPTS

- Processes
 - Instances of a computer program
- Threads
 - Statements executed by a scheduler
- Pre-emptive multitasking
 - Each thread is given a finite amount of time (time-slice)



Thread-Safety

4



THREAD-SAFETY

- Testing cannot prove thread-safety
 - After ton of testing, all you can say is that your code is probably thread-safe
- At most, testing can prove thread-(un)safety

Only “logic” can prove thread-safety



DATA THREAD-SAFETY

- Where it will be used?
 - Only Main Thread?
 - Main Thread + other threads?
 - Only secondary threads?
- What type of data?
 - It is mutable?
 - It is protected (the access)?



THREADS & UI (VCL/FMX)

- Usually the UI (VCL/FMX) is not thread-safe
 - The UI is in the Main Thread
- Never access the UI from a secondary thread
 - Decouple threads processing and UI updates



Synchronization

5



CONCEPTS

- Access to the same data from different threads
 - You need to synchronize that access
- The simplest way is to “lock” the resource while using it
 - Other threads will get in queue waiting for the resource to become accessible
- Difference between read and write operations



CONCEPTS

- Atomic operations
 - Non-atomic operations
- Thread-safe means convert non-atomic operations in non-interruptible operations
- `Unit System.SyncObjs`



Demo:

TSafeList Example

- ★ Create/use a thread-safe list
- ★ The all the code that uses the list is thread-safe?



BUT... I DON'T USE THREADS

- Have you ever used a communication server?
 - TIdHttpServer, TIdTCPServer
- REST server's methods
 - WiRL, RADServer
 - DataSnap (TCP or REST)
- **The same rules still apply**



Demo:

Communication Server Example

- ★ Use of TIdHttpServer
- ★ Handle the OnCommandGet
- ★ Calling the event from a Button: differences?



WHAT IS DATA?

- Variables
 - Local
 - Global
- Objects
 - Fields/Properties
- DataSets
- DataModules
- Forms



SYNC DATA ACCESS

- Only 1 thread at time can access the data (R/W)
- If you fail to synchronize
 - Race Conditions
 - Data Inconsistencies
 - Deadlocks



DATA (RESOURCES) SYNC

- Synchronize access to a global object/var between threads
- Synchronization is essential for
 - Data Integrity
 - Program Correctness
 - Preventing Undefined Behavior
- Synchronization is achieved through “synchronization objects”



System.SyncObjs

- TCriticalSection
- TMonitor
- TMutex
- TEvent
- TSemaphore
- TSpinLock
- TMultiReadExclusiveWriteSynchronizer
- ...



Sync Objects Comparison

Unsafe	110 ms
TCriticalSection	141 ms
TMonitor	234 ms
TMutex	1422 ms
Named TMutex	1426 ms
TEvent	1340 ms
TSemaphore	1407 ms
TSpinLock	171 ms

Compiler: Delphi 10.2.3 (Tokyo)



Demo:

Data Access Synchronization

- ★ Using a TCriticalSection
- ★ Using TMonitor
- ★ Using a TEvent



MAIN THREAD SYNC

- How to synchronize with the main thread (UI)?
 - Call the Synchronize() method
- How Synchronize() works?
 - Execute in the main thread's context the procedure passed as param
 - Let's draw a diagram
- What to put in the Synchronize method
 - The less code the better
- TThread.Queue (better way)



Demo:

Synchronize to the Main Thread

- ★ Update a label in the OnCommandGet (TIdHttpServer)
- ★ Example with a secondary thread



Legacy Code

5



WHERE IS THE CODE??

- Good
 - Forms: 30%
 - DataModules: 40%
 - Custom Classes: 30%
- Bad
 - Forms: 60% (+)
 - DataModules: 30%
 - Custom Classes: 10%



DATAMODULES

- Remove **uses** of forms and UI code!
 - uses FormXYZ
- Remove “application” events, state management, etc...
 - At least move them to another DM
- Encapsulate SQL code in ready to use functions
 - function GetOrdersCount(): Integer
 - Don't use global queries/datasets but local variables



Demo:

DataModule Refactoring (1)

- ★ Move “application” code to another DM
- ★ Transform a query result in a function



DATASETS & UI

- DataSet + DataSource + Grid
 - The Delphi sacred triad!
- Can you leave them in the DataModule?
 - Yes, but...
 - All DataSets closed at DesignTime
 - Don't open them in the DataModuleCreate
- The best scenario is to have separated DM for “design-time” datasets



Demo:

DataModule Refactoring (2)

- ★ DataSets used at design-time
- ★ Where to put the TDataSource?
- ★ The TDataSource events



CONNECTIONS

- 1 connection per thread
- How to achieve that?
 - Use a new connection for every query (request)
 - Use a connection pool
 - Create a new connection for a new query
 - In FireDAC there is one ready to use
 - Create a DataModule instance per thread



Demo:

DataBase Connections

- ★ Create a new connection per request
 - `Query.Connection := Connection1 → Query.Connection := NewConnection()`
- ★ Using the connection pool *
- ★ Create a DM instance per thread



SYNCRONIZE TO THE UI

- Why is it more difficult to sync with the UI
 - VCL/FMX Thread
 - OS Messaging
- How to call Synchronize()
- Synchronize() or Queue() ?



Demo:

UI Synchronization

- ★ Call to Synchronize()
- ★ Call to Queue()
 - ForceQueue()
- ★



SYNCHRONIZE TO DATA

- Why do I have to “protect” the access to global objects?
- When an “object/variable” is considered global?
- Where do I have to put the protection code?



CRITICAL SECTION

- TCriticalSection in unit System.SyncObjs
- Lock access to the critical section (hence the name) of code
- Steps to use it
 - Define a TCriticalSection (scope matters)
 - Create a CriticalSection
 - Acquire a CriticalSection
 - Leave a CriticalSection



MONITOR

- System.TMonitor is a record
- Like a TCriticalSection plus a condition variable
- Use only if necessary (IMHO)
- Bugs and performance problems in early XE versions



MUTEX

- **M**utually **E**xclusive
- Like a `TCriticalSection`
- Used for inter-process communication



SEMAPHORE

- Like a TMutex
- You can limit access on more threads
- Access pooling (sort of)



EVENT

- Like a Mutex
 - Acquired, Released
- An event is used to signal a thread
- Used (also) to wait in the thread execution



INTERLOCKED OPERATIONS

- Static class TInterlocked (System.SyncObjs)
 - Increment, Add, Exchange, CompareExchange
- Locking is done by the CPU
- Mapped on Atomic* function in System



Demo:

Data Synchronization

- ★ TCriticalSection
 - Where to declare the CS
 - Enter, Exit
- ★ Using TMonitor



THANK YOU