iTDEVCON 2024
EUROPEAN DELPHI CONFERENCE

Do you know how to debug?

Primož Gabrijelčič

ROME - NOVEMBER 14th, 15th 2024

About me

Primož Gabrijelčič

http://primoz.gabrijelcic.org

- programmer, MVP, writer, blogger, consultant, speaker
- Blog *https://thedelphigeek.com*
- Twitter *@thedelphigeek*
- Skype *gabr42*
- LinkedIn *gabr42*
- GitHub *gabr42*
- SO *gabr*

The Delphi Geek

random ramblings on Delphi, programming, Delphi programming, and all the rest

Friday, April 05, 2024

Delphi and Python, working happily together

Next Wednesday, 10th, I'll be talking about Delphi and Pyhon in Ljubljana. As usual for Slovenian workshops, the talk will be in Slovenian language so I'll continue this invitation appropriately ...

Python je izjemno priljubljen programski jezik, ki slovi po svoji raznoliki in preprosti uporabi. Skupnost razvijalcev nenehno prispeva k njegovemu razvoju in izboljšanju jezika, kar dodatno krepi njegovo privlačnost za širok nabor uporabnikov. Kot večina večplatformnih sistemov pa ne slovi po podpori za izdelavo uporabniških vmesnikov.

Tu nastopi **Delphi**, ki slovi po svoji hitrosti in učinkovitosti in je odličen za razvoj aplikacij za Windows okolje. Zato ni presenetljivo, da okolji ne delujeta kot neposredna konkurenca, pač pa se izredno dobro **dopolnjujeta**.

Največja prednost je združevanje moči obeh jezikov. Delphi ponuja hitrost in zmogljivost, Python pa berljivost in obsežno knjižnico razširitev. **S tem lahko razvijalci izkoristite najboljše iz obeh svetov.**

Na predavanju si bomo ogledali:

- Kako lahko Python in Delphi sodelujeta na dva različna načina,
- predstavili bomo orodji DelphiVCL in DelphiFMX, s katerima lahko v Pythonu enostavno ustvarite lep uporabniški vmesnik
- ter nekaj orodij (Python4Delphi, PythonEnvironments, Lightweight Python Wrappers), ki omogočajo, da kodo, napisano v Pythonu, vgradite v svoj Delphi program.

Spotoma si bomo na kratko ogledali še novosti v novem **Delphi 12.1**!

Kliknite za prijavo!

Posted by gabr42 at 09:39 No comments:

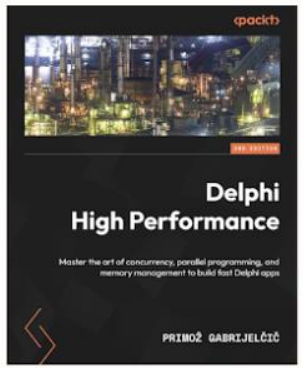
Labels: Delphi, presentations, Python

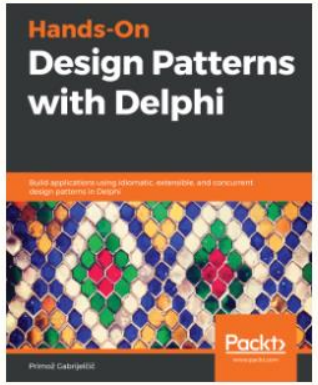
Saturday, December 02, 2023

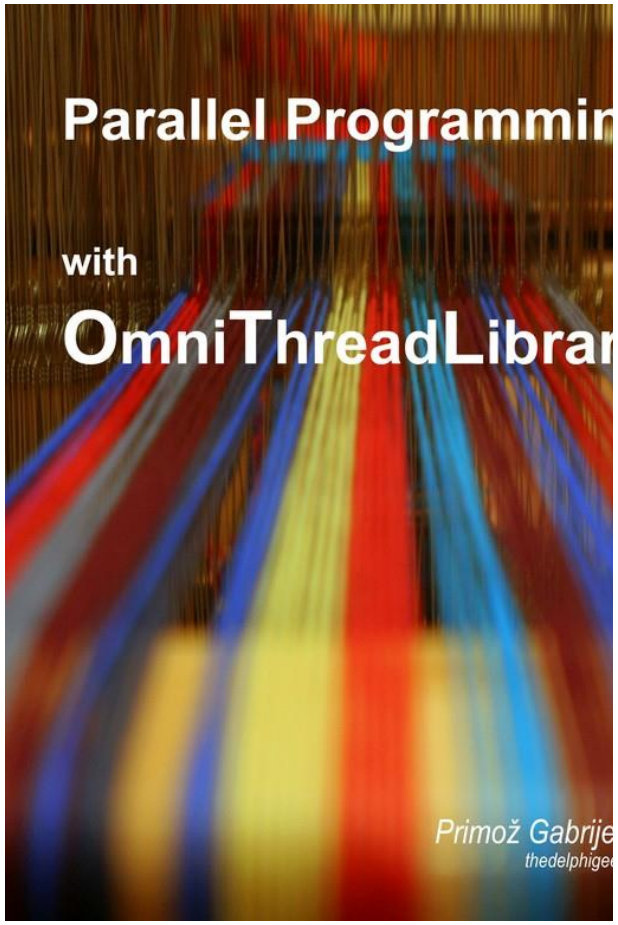
embarcadero MVP

Pages

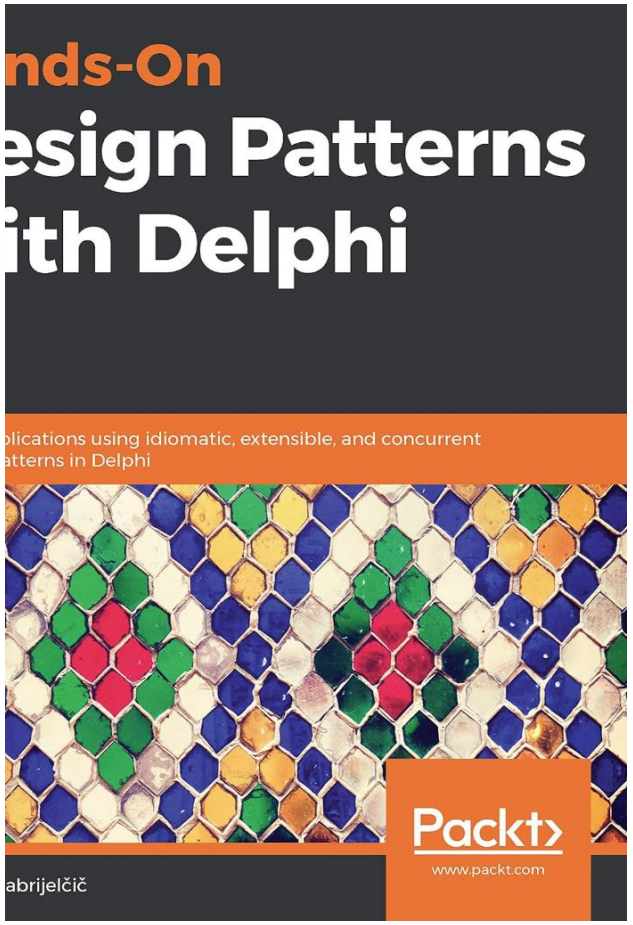
Presentations







https://delphi-books.com/



“Debugging is a **methodical** process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus making it behave as expected.“

--unknown

Be methodical!

Debugging loop

1. Make sure you can repeat the problem!
    - If a problem appeared elsewhere, you may need a correct configuration
    - Or a correct version of the code from the repository
    - Or it could be a random problem that is hard to repeat
2. Establish a hypothesis
    - You can start with “I have absolute no idea what is going on”
3. Gather data
    - Step through code, use breakpoints, inspect data
    - Logging
4. Implement potential fix
5. Test
    - See item 1
6. If not fixed, go to 2

1. Repeat the problem!

- Bugs can depend on software configuration …
- ... Or on a specific input data
- ... Or on specific version of your code
  - If you’re lucky, the bug was already fixed
- … Or on an OS version
- … Or on a specific hardware
- ... Or, especially in multithreaded code, on random chance

1. Repeat the problem (there's more to say!)

- If the problem is hard to reproduce, try
  - Stress-testing
    - Slow CPU, multiple clients
    - Randomly pause program in debugger and resume after short time
  - Reproduce exact input data from the customer's side
    - Create special version that records all the data there
    - Reproduce on testing computer
  - Test on multiple computers
  - Debug directly on problematic installation (Remote Debugger)

2. Establish a hypothesis

- If you have a rough idea on what could be going wrong, try that first ☺
- Don’t focus on a single idea too much as it may be completely wrong!
  - Frequently you’ll be completely wrong
- Test multiple hypotheses in parallel

3. Gather data

- Pause the program (Breakpoints)
- Inspect variables (Evalutate, Watch)
- Single step through the problematic code
  - Again, you may not be looking at the real reason for the problem!
- Repeat

- Automate that with logging
  - Especially important when debugging multithreaded problems
  - Pause + step + evaluate may change program behaviour
- OutputDebugString, GpStuff.OutputDebugString
- Console output, GpConsole
  - Fast!
- Any logging suite
  - LoggerPro, CodeSite, Log4Delphi
- Capture exception stack on problematic computer
  - MadExcept, EurekaLog, JclDebug

4. Implement potential fix

- You can only assume it will work

5. Test

- Testing = failure to repeat a problem
- This is a problem especially in multithreaded code
  - Where problem may just “hide away” because of the fix
- You’ll maybe have to revisit the problem at the later stage
- When documenting (commit log) always state what you **believe** was the cause of the problem

Know the tools

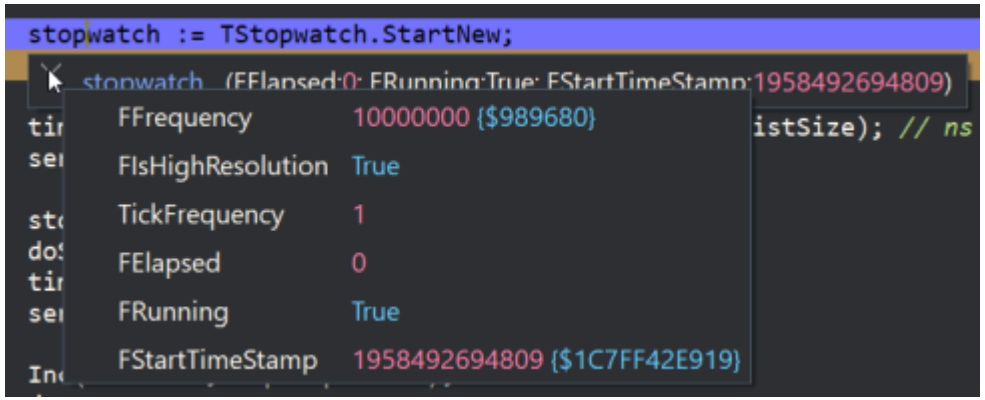
Basic shortcuts

- Ctrl+F2 Program reset
- F4 Run to cursor
- F5 Toggle breakpoint
- F7 Execute next line, step into funtions
- F8 Execute next line, step over functions
- Shift-F8 Execute until return from function
- F9 Run with debugging

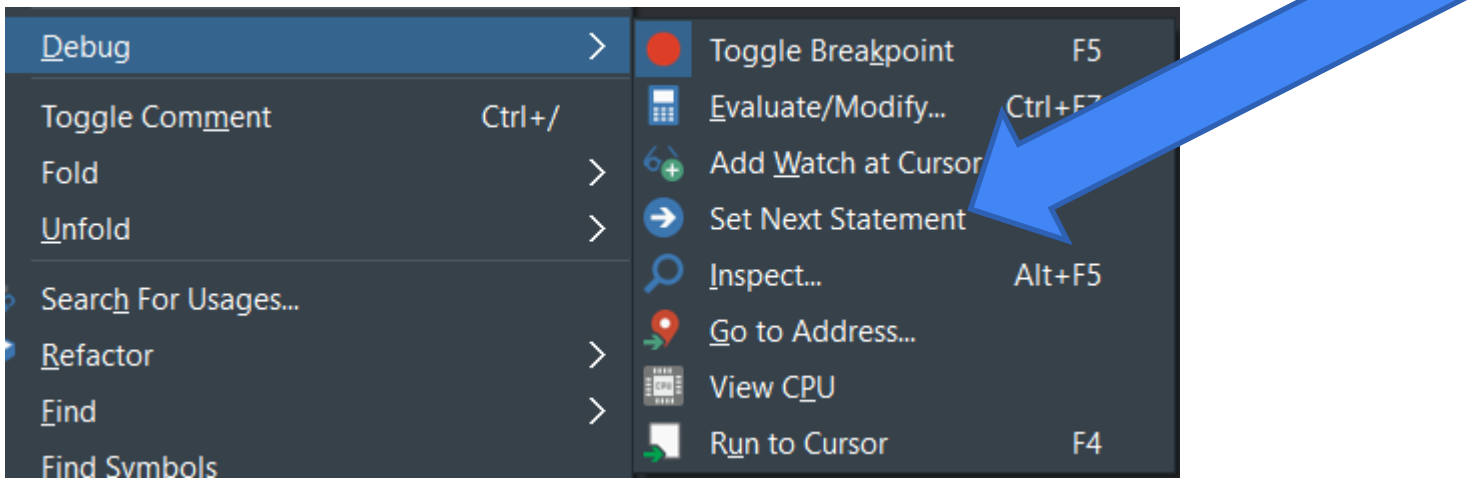
- Ctrl+F5 Add watch
- Alt+F5 Inspect
- Ctrl+F7 Evaluate/Modify

Debug with the mouse

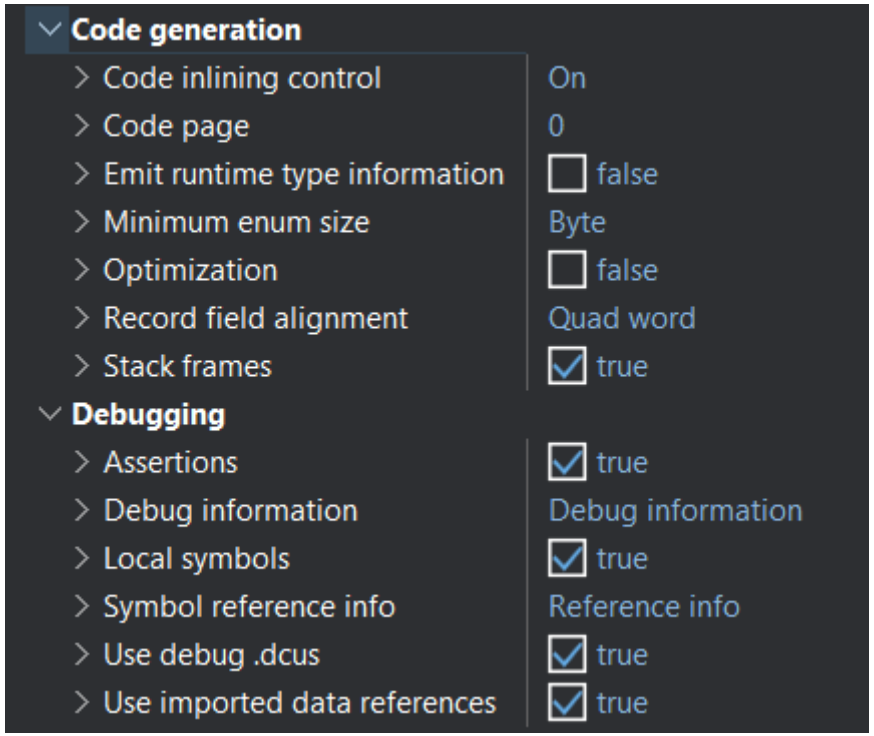
- Ctrl-Shift-<click> Inspect
- Mouse hoover Evaluate

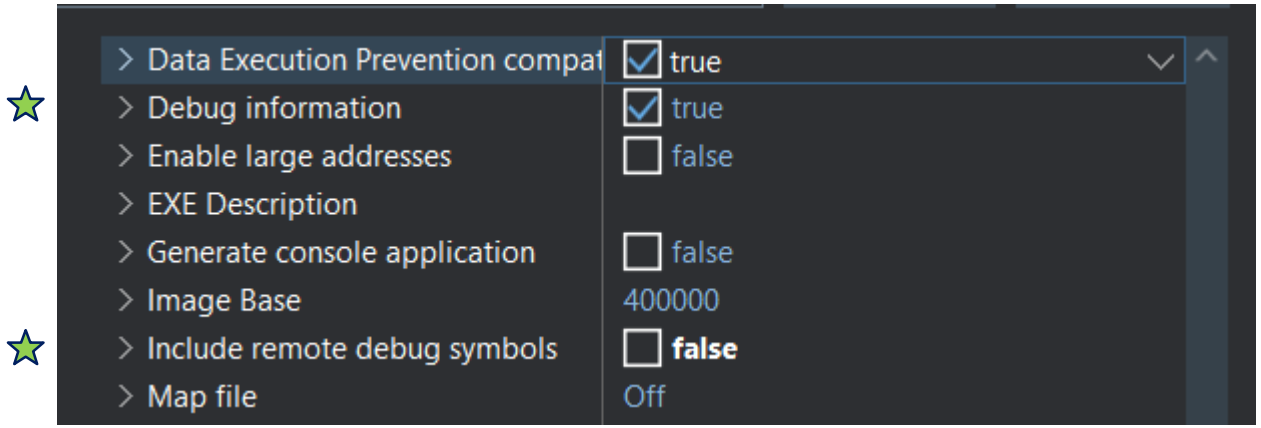


- <Right-click>, Debug

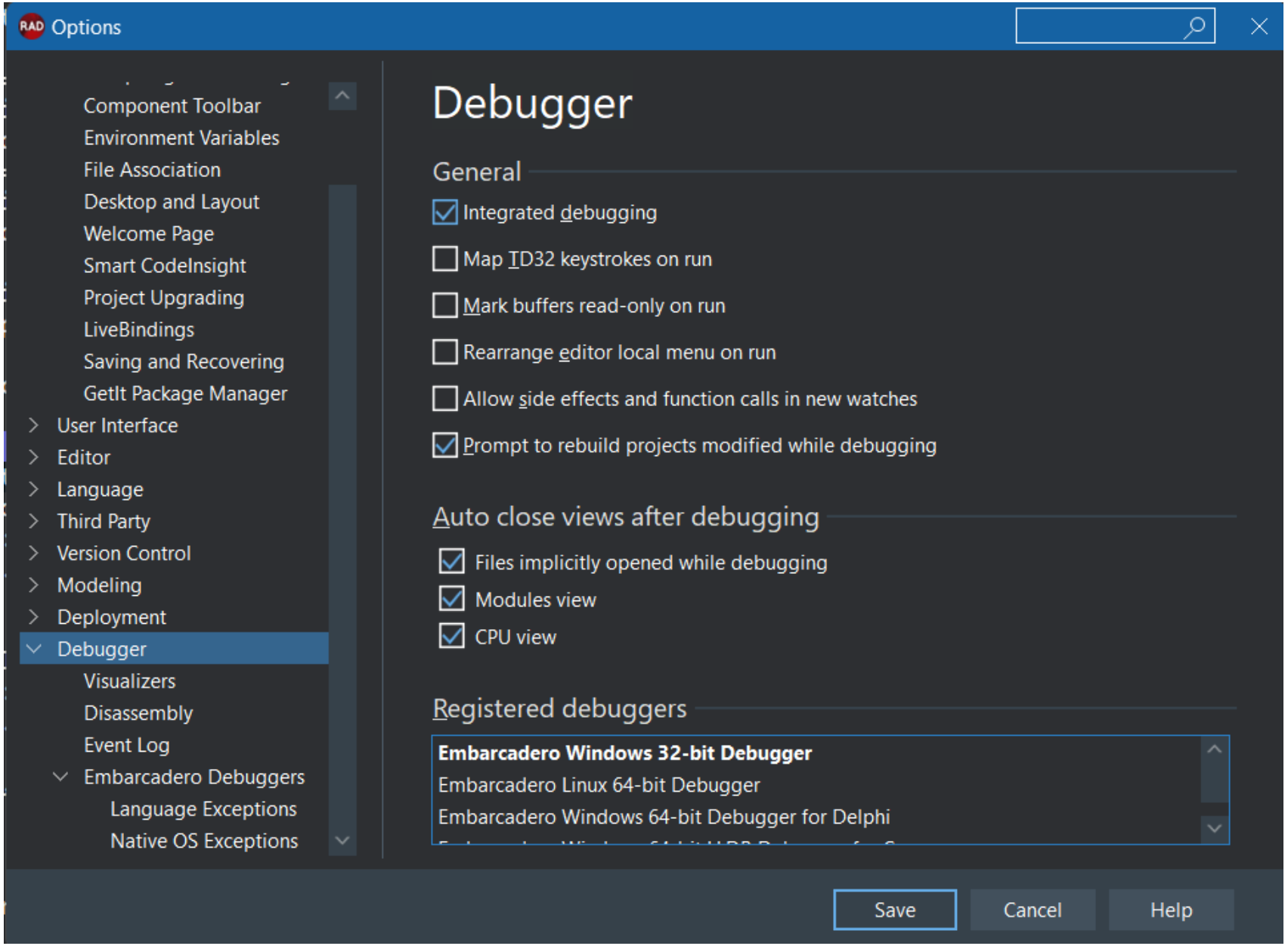


Project and Linker options





Tools, Options, Debugger



Tips & tricks

Conditional breakpoints

- Set conditional breakpoitns
  - Breakpoint properties
    - Condition
    - Pass count
    - Thread
    - Group
    - Enable group, Disable group
  - Slow!

- Conditional breakpoints in code

```
if (condition) then
  sleep(0); // put breakpoint here
```

- Or use GpStuff:

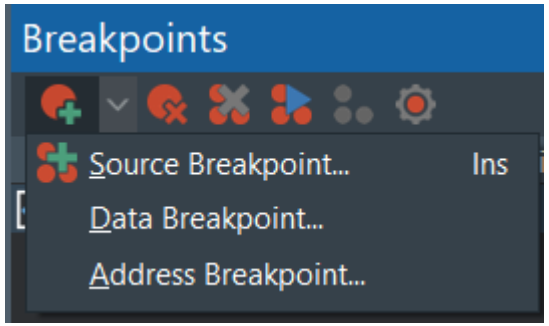
```
DebugBreak(condition);
```

High pass count trick

- Create a breakpoint
- Set a high pass count (9999999)
- Run the program until it crashes
- Read the current pass count
- Change pass count to that value
- Rerun

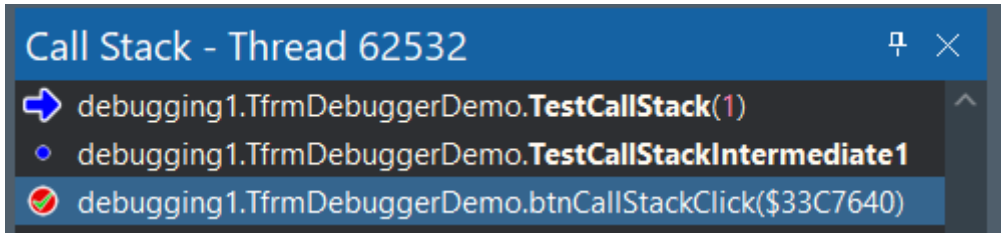
Hardware breakpoints

- Implemented in the CPU
- Address breakpoint
  - When code at address is executed
- Data breakpoint
  - When data at address if modified

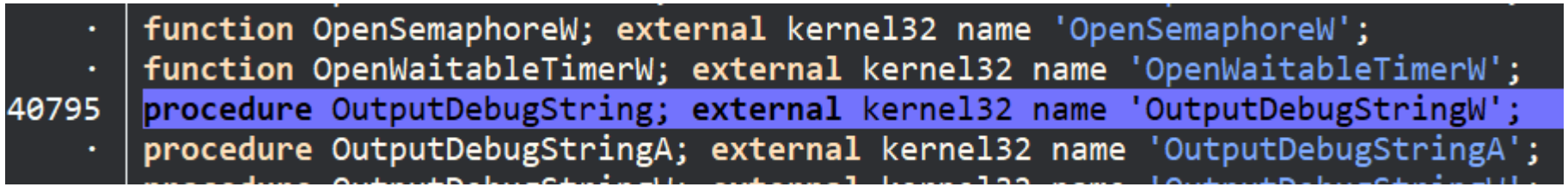


Breakpoint tricks

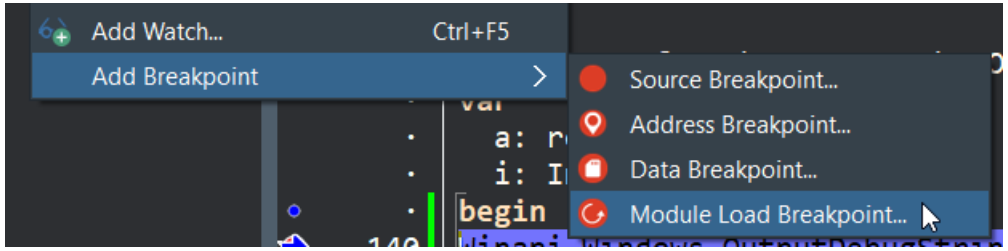
- Breakpoint can be set on the call stack
  - Triggers when the code returns to that point



- Breakpoint can be set on a function import
  - Requires “Use debug .dcus”



- Breakpoint can be set on module load



Looking for problematic ‘initialization’

- Open System unit
- Find procedure InitUnits
- Use “High pass count” trick to find problematic initialization section

Use breakpoints for logging

- Breakpoint properties
  - Break (disable)
  - Evaluate expression
  - Log message
  - Log call stack
- Combine with other breakpoint properties
  - Conditional evaluation etc

Logging

- Make output more visible in debug log

```
OutputDebugString(#13#10'test'#13#10);
```

- Conditional logging

```
if (condition) then
  Log(some_stuff);
```

- On demand

```
var b := false;
if b then
  Log(some_stuff);
```

Log to console

```
uses GpConsole;
Console.Writeln('xxx');
Console.Write(['The value is ', value]);
```

- Fast
- Thread-safe
- Can output simple data types (incl. Pointers and Booleans)
- Can be used in VCL, FireMonkey, and Console applications
- Disable in one line

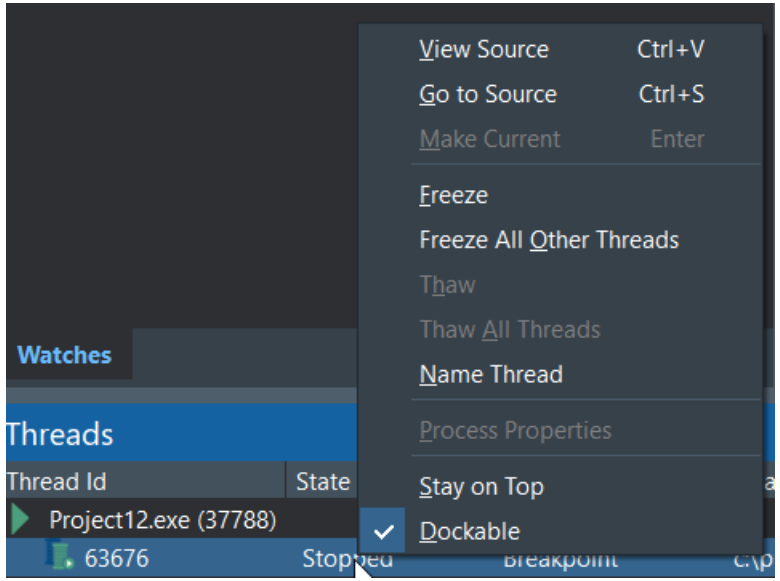
  ```
  Console.Disabled := true;
  ```

- Use colors to enhance readability

  ```
  Console.Writeln(['And the answer is {bright red on bright yellow}', 42, '{}!']);
  ```

Debugging multithreaded programs

- Threads window
- Right-click
  - Freeze
  - Freeze All Other Threads
  - Thaw
  - Thaw All Threads



Debugging multiple programs

- Open project group
  - **Build all**
- Select first program
  - **Run**
- Select second program
  - **Run**
- Debugger is attached to all running instances

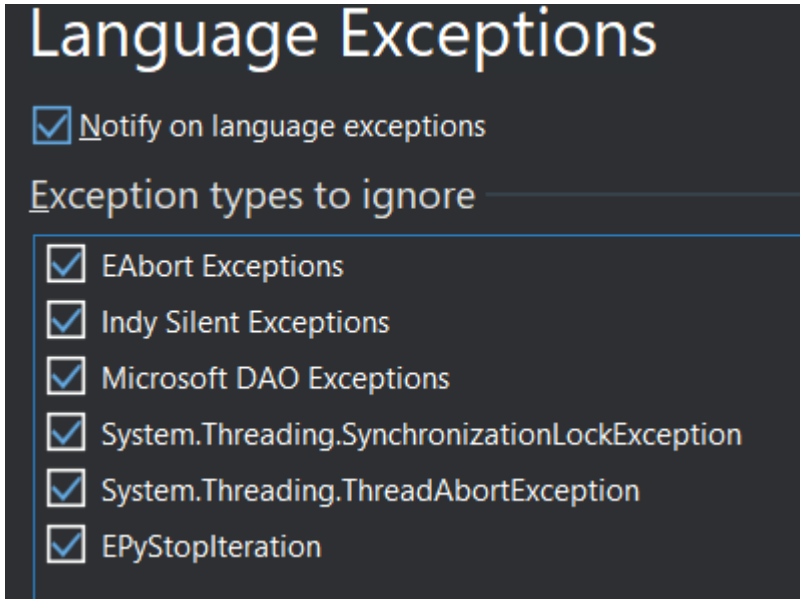
Switching between programs

- Run, Attach To Process
- Run, Detach From Program

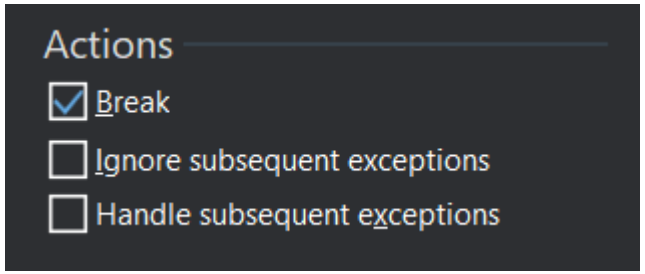
- Sometimes unstable (can crash)

Ignore exceptions

- Tools, Options, Debugger, Embarcadero Debuggers, Language Exceptions



- Or wrap code in two breakpoints
  - First disables exceptions
  - Second re-enables exceptions



- Or use GpStuff

```
var ignore := ExceptionsInDebugger.Ignore;
...
ignore := nil;
```

GpXXXX units

https://github.com/gabr42/GpDelphiUnits