A DSL for driver development, why & how?

Louis Opter
www.rathaxes.org

Fosdem 2013
About us

- Three guys, limited experience;

Lionel's idea back in 2007;
Experimentations, POC in 2009;
A side project for us since 2010.
About us

- Three guys, limited experience;
- Lionel’s idea back in 2007;
About us

- Three guys, limited experience;
- Lionel’s idea back in 2007;
- Experimentations, POC in 2009;
About us

- Three guys, limited experience;
- Lionel’s idea back in 2007;
- Experimentations, POC in 2009;
- A side project for us since 2010.
Why?

- Learn;
- Biggest part of OSes code base;
- Dominant cause of crashes;
- Complexity/Skills required;
Why?

- Learn;
- Biggest part of OSes code base;
- Dominant cause of crashes;
- Complexity/Skills required;
- *Understanding Modern Device Drivers* [1].
Why?

- Learn;
- Biggest part of OSes code base;
- Dominant cause of crashes;
- Complexity/Skills required;
- *Understanding Modern Device Drivers* [1].

How can this be solved/improved?
Pre-requisites

- **Kernel**: The Base Component of OSes;
- **Device**: The hardware on the motherboard;
- **Subsystem**: Abstraction layer for big features (Network, Storage, USB...);
- **Driver**: The software component loaded into the kernel;
- **Paradigm**: Monolithic, Exo, Hybrid, Micro.
ToC

1. “State of the art”;
2. Quick dive into Rathaxes;
3. Wrap-up & next steps.
“State of the art”
SOA - RAD Tools

Jungo’s WinDriver: http://youtu.be/-o6M11jZMQk
SOA - Static Analysis

- Static Driver Verifier (Windows specific);
- Safe Drive (Linux) [2];
- SymDrive [3].
SOA - Static Analysis

Limitations:

- Doesn’t solve code re-usability;
- SymDrive is very interesting.

Cover topics we don’t cover yet;
Devil [4];
Hail (similar to Devil).
SOA - Interface Description Languages (IDL)

- Describe interfaces to access the registers with constraints;
- No notion of bus (in Devil);
- Don’t describe the algorithms to access the registers.
Only solve one part of the problematic;

Nonetheless, Rathaxes started from Devil.
SOA - Domain Specific Languages (DSL)

- Termite [5];
- Us: Rathaxes.
SOA - DSL - Termite

- Generate a complete driver as a FSM;
- Sources are three “specifications”: 

SOA - DSL - Termite

- Generate a complete driver as a FSM;
- Sources are three “specifications”:
  - device-class specification;
  - device specification;
  - os specification.
Limitations:

- Interface with the OS is very blurry:
  - The FSM is OS agnostic;
  - Model of “messages”.
- FSM generation can take hours;
- Only the paper is available, no code.
Quick dive in our WIP
WIP - How we see drivers

What we have:

- Device dependent registers;
- Device dependent logic;
- Kernel dependent subsystems.
WIP - Rathaxes - Model

- Describe driver & kernel in Rathaxes;
- Generate driver in C;
- Sources are split in three parts:
Describe driver & kernel in Rathaxes;
Generate driver in C;
Sources are split in three parts:
▶ BLT: kernel dependent subsystems;
▶ RTI: interfaces;
▶ RTX: device dependent registers and logic (ideally).
Describe driver & kernel in Rathaxes;
Generate driver in C;
Sources are split in three parts:
  ▶ BLT: C code templates;
  ▶ RTI: interfaces;
  ▶ RTX: our DSL front-end.

Loosely similar to Termite’s specifications.
WIP - Some code - ethernet.blt

Ethernet initialization code (link)
WIP - Debrief

- Re-usability is achieved;
- Device dependent algorithms can be extracted;
- Interfaces describes subsystems and needed device functionality;
- Interfaces are paradigm agnostic.
Wrap-up & next steps
Wrap-up

- Research is in progress by different teams;
- Research seems justified;
- Our DSL front-end is not implemented yet;
- We are WIP but we are the only open source project.
Next steps

- Complete that e1000 example;
- Mature the language;
- Start to implement the front-end;
- Get students on the project;
- Move the compiler to Python.
Questions? Feedback?

Thanks

► http://www.rathaxes.org/
► #rathaxes on IRC (chat.freenode.net)
► @rathaxes
CNorm 3 / CodeWorker

CNorm & CodeWorker are used to write the Rathaxes compiler.

CNorm 3.x is:
- A C Frontend for an Epitech school’s project;
- Written in the CodeWorker scripting language;

CodeWorker is:
- Allow to parse complex things (like ANTLR, Bison...);
- Easy for a one semester project;
- Have some features that other tools don’t have.
CNorm 4 / Pyrser

Pyrser is:
- A Python package;
- Have same features than codeworker;
- Release soon (needed for CNorm 4).

CNorm 4.x is:
- The Rewrite of CNorm in Python with Pyrser;
- Must be available for the next school semester;
- Going to be used to rewrite the Rathaxes compiler.
Bibliography


