A DSL for driver development, why & how?

Louis Opter
www.rathaxes.org

Fosdem 2013

About us

- Three guys, limited experience;

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/-o6M1ljZMQk`

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*;

SOA - Interface Description Languages (IDL)

- 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.

SOA - Interface Description Languages (IDL)

- 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.

SOA - DSL - Termite

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:

WIP - Rathaxes - Model

- 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).

WIP - Rathaxes - Model

- 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 functionnality;
- 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 langage;
- 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

[1] Asim Kadav and Michael M. Swift.
Understanding Modern Device Drivers.
In *ASPLOS*, London, UK, March 3-7 2012.
http://pages.cs.wisc.edu/~kadav/study/.

[2] Feng Zhou, Jeremy Condit, Zachary Anderson, and Ilya Bagrak; Rob Ennals; Matthew Harren, George Necula, and Eric Brewer.
Safedrive: Safe and recoverable extensions using language-based techniques.
In *OSDI '06*, Seattle, WA, November 6-8 2006.
http://ivy.cs.berkeley.edu/safedrive/.

[3] Matthew J. Renzelmann, Asim Kadav, and Michael M. Swift.
Symdrive: Testing drivers without devices.
In *OSDI '12*, Hollywood, CA, October 8-10 2012.
http://research.cs.wisc.edu/sonar/projects/symdrive/index.shtml.

[4] L. Réveillère.
Approche langage au développement de pilotes de périphériques robustes.
Thèse de doctorat, Université de Rennes 1, France, December 2001.
http://www.labri.fr/perso/reveille/publications/papers/reveillere-thesis.pdf.

[5] L. Ryzhyk, P. Chubb, I. Kuz, E. Le Sueur, and G. Heiser.
Automatic device driver synthesis with Termite.
In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, Big Sky, MT, USA, Oct 2009.
http://ssrg.nicta.com.au/publications/papers/Ryzhyk_CKSH_09.pdf.