LUA

Lukas Niederstätter
VU Specialization Seminar, 30.06.2023

Content

- Introduction
- History
- Concepts and Versions
- Use-Cases
- Conclusion
- Final Thoughts

Introduction

Introduction

- powerful and lightweight
- derived from C/C++ and SOL
- many uses in wider areas

[1]

History

History

- founded 1993

[1]

- developed for intern programs

- combination of its predecessors

[2]

DEL

(Data Entry Language)

- entries and predicate logic
- basic “properties”
- compared to records in other languages

```
:e      gasket                  "gasket properties"
mat     s                        # material
m       f       0                # factor m
y       f       0                # settlement stress
t       i       1                # facing type

:p
gasket.m>30
gasket.m<3000
gasket.y>335.8
gasket.y<2576.8
```

DEL code snippet [3]

SOL

(Simple Object Language)

- introduction of tracks
- multiple object types via typa declaration
- derived from BiBTex and UIL

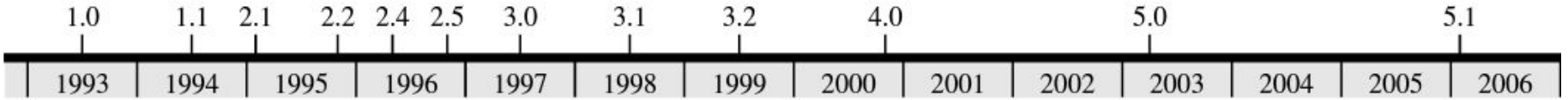
```
-- defines a type `track', with numeric attributes `x' and `y',
-- plus an untyped attribute `z'. `y' and `z' have default values.
type @track { x:number,y:number= 23, z=0}

-- defines a type `line', with attributes `t' (a track),
-- and `z', a list of numbers.
-- `t' has as default value a track with x=8, y=23, and z=0.
type @line { t:@track=@track{x=8},z:number*}

-- creates an object `t1', of type `track'
t1 = @track { y = 9, x = 10, z="hi!"}

-- creates a line `l', with t=@track{x=9, y=10},
-- and z=[2,3,4] (a list)
l = @line { t= @track{x=t1.y, y=t1.x}, z=[2,3,4] }
```

SOL Code - Track Declaration [3]

Concepts & Early Versions

Concepts & Early Versions

- early versions still inspired by SOL
- functionalities were quick to follow
- more derivations



[5]

	1.0	1.1	2.1	2.2	2.4	2.5	3.0	3.1	3.2	4.0	5.0	5.1
libraries	4	4	4	4	4	4	4	4	5	6	8	9
built-in functions	5	7	11	11	13	14	25	27	35	0	0	0
API functions	30	30	30	30	32	32	33	47	41	60	76	79
vm type (stack × register)	S	S	S	S	S	S	S	S	S	S	R	R
vm instructions	64	65	69	67	67	68	69	128	64	49	35	38
keywords	16	16	16	16	16	16	16	16	16	18	21	21
other tokens	21	21	23	23	23	23	24	25	25	25	24	26

Lua - Releases and Updates [4]

	1.0	1.1	2.1	2.2	2.4	2.5	3.0	3.1	3.2	4.0	5.0	5.1
constructors	●	●	●	●	●	●	●	●	●	●	●	●
garbage collection	●	●	●	●	●	●	●	●	●	●	●	●
extensible semantics	○	○	●	●	●	●	●	●	●	●	●	●
support for OOP	○	○	●	●	●	●	●	●	●	●	●	●
long strings	○	○	○	●	●	●	●	●	●	●	●	●
debug API	○	○	○	●	●	●	●	●	●	●	●	●
external compiler	○	○	○	○	●	●	●	●	●	●	●	●
vararg functions	○	○	○	○	○	●	●	●	●	●	●	●
pattern matching	○	○	○	○	○	●	●	●	●	●	●	●
conditional compilation	○	○	○	○	○	○	●	●	●	○	○	○
anonymous functions, closures	○	○	○	○	○	○	○	●	●	●	●	●
debug library	○	○	○	○	○	○	○	○	●	●	●	●
multi-state API	○	○	○	○	○	○	○	○	○	●	●	●
`for` statement	○	○	○	○	○	○	○	○	○	●	●	●
long comments	○	○	○	○	○	○	○	○	○	○	●	●
full lexical scoping	○	○	○	○	○	○	○	○	○	○	●	●
booleans	○	○	○	○	○	○	○	○	○	○	●	●
coroutines	○	○	○	○	○	○	○	○	○	○	●	●
incremental garbage collection	○	○	○	○	○	○	○	○	○	○	○	●
module system	○	○	○	○	○	○	○	○	○	○	○	●

[4]

DEMO

USE-CASES

Use-Case: Web

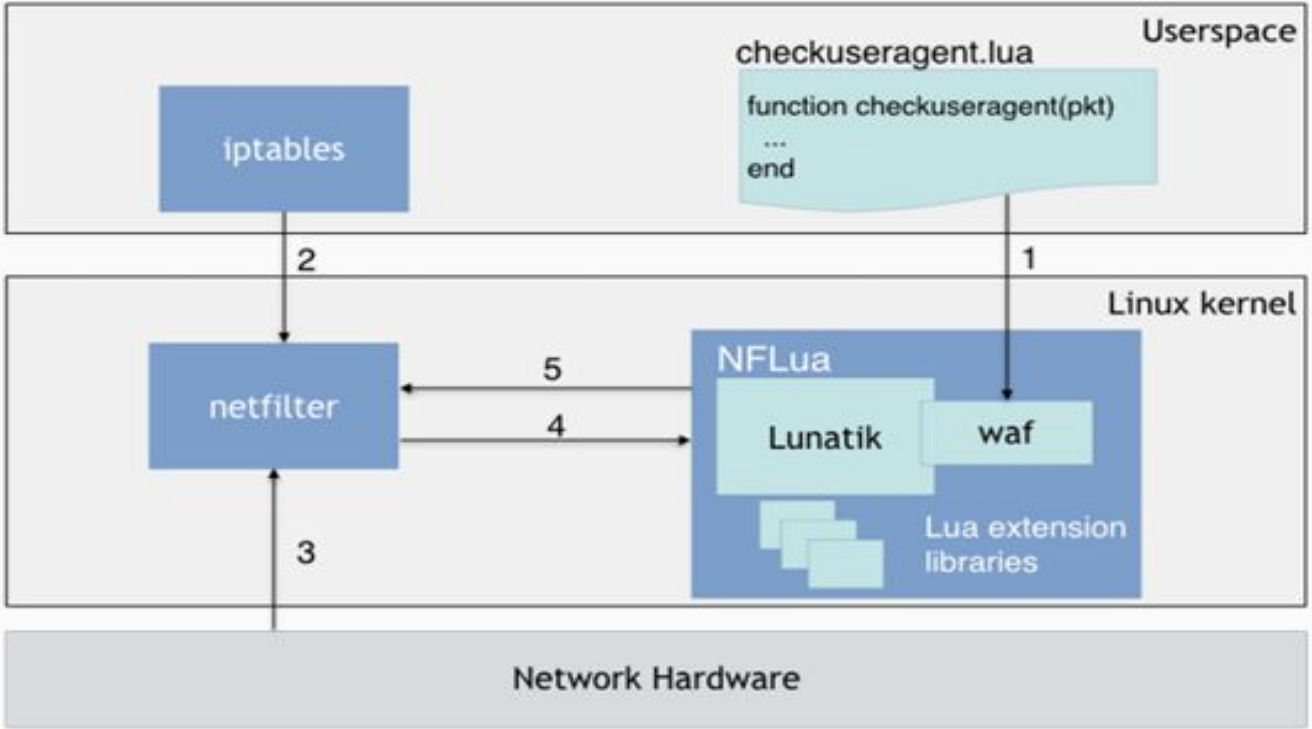
- CGILua for web development
- flexible, extensible web pages
- applying standards easier

Use-Case: Embedded Systems

- integrated in handheld devices
- allowing warning-free compilation
- used in embedded processors

Embedded Systems

- telephony-network testing
- scripts for microprocessors
- ethernet switches



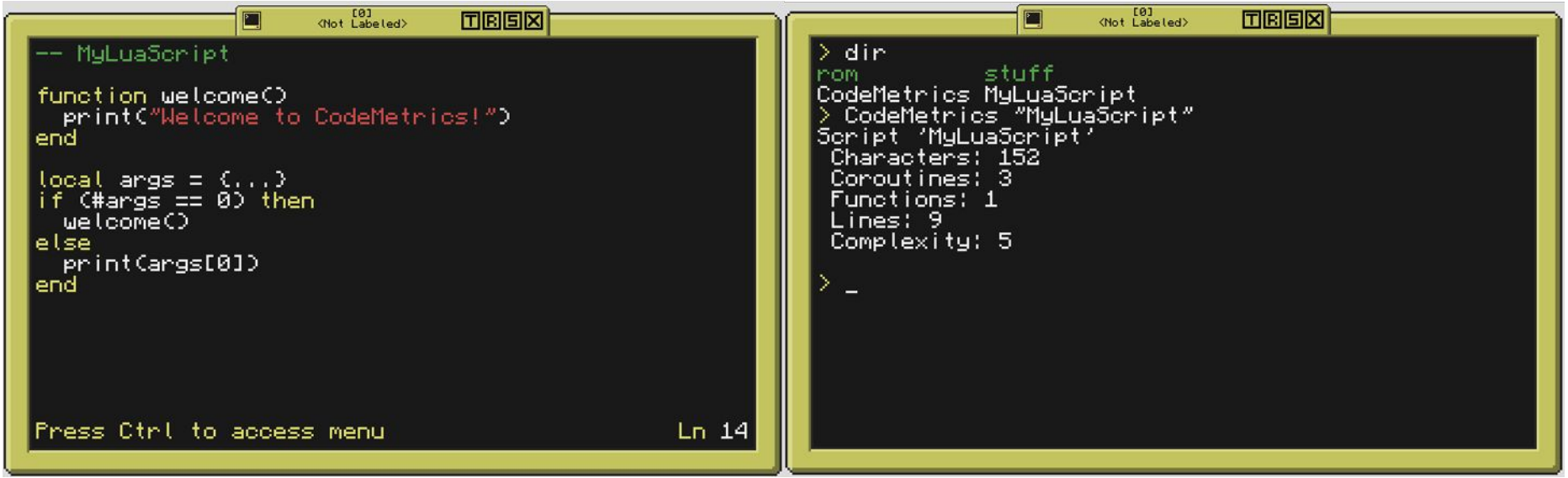
Netfilter with Lua, NFLua [5]

Use-Case: Game Development

- Usage with embedded script with JITLua
- short learning curve and easy to integrate
- 2003: Most popular language for its use-case

Game Development

- Warframe: HUD and GUI Scripts
- Roblox: Using Luau in game engine
- Minecraft: Providing modifications and in-game coding



Coding inside Minecraft [6]

Conclusion

Conclusion

- well-established community in terms of updates and use
- widespread usage in many different sectors
- applying newer versions to older proves to be difficult
- success especially in game development

Final Thoughts

Can Lua Be Replaced?

- although very efficient, relying on C API
- newer versions and modifications of Lua based on Python, C
- Lua draws connections to Modula, C, Tex
- Other languages more suited (e.g. Web-Development with JavaScript)

Should Lua be replaced?

- very good language for embedded systems
- could be replaced in web
- game modifications and development easier with Lua

Lua in the future...

- potential to influence even more technical sectors
- increased use in game development
- backwards compatibility allows easier web development

Sources

- [1]: https://www.lua.org/authors.html, Lua Authors
- [2]: https://www.tecgraf.puc-rio.br/, Tecgraf/Petrobras
- [3]: https://www.lua.org/history.html History of Lua
- [4]:https://dl.acm.org/doi/abs/10.1145/1238844.1238846?casa_token=Kj0SBM_gdHoAAAAA:2vlmUsP07ZPF0mQXCLdmLKhN6U7SlyaHl2MJPce-_gKbkwfPV5CpDjwCq6Xzy5UC_U0oDy5C7w Inproceeding: Evolution of Lua
- [5]:https://netdevconf.info/0x14/pub/slides/22/netscript-presentation-200805.pdf, Network Scripting with Lua
- [6]: https://ocdoc.cil.li/, Lua Embedded Coding in Game Modification

Thank You for Your attention