Ein neuer Motor für ConTEXt

Henning Hraban Ramm

Seit über drei Jahren konzentriert sich die ConTEXt-Entwicklung auf die neue »Engine« LuaMetaTEX.

Zeitplan

Vor fast vier Jahren entstand die Idee zu LuaMetaTEX und fand breite Zustimmung unter den ConTEXt-Anwendern, denen sie beim ConTEXt Meeting 2018 präsentiert wurde. Anfang 2019 stellte Hans Hagen die erste Testversion vor, und beim Meeting 2019 die erste offizielle Version.

Schon beim Meeting 2020 hatte er mehr oder weniger erreicht, was er sich vorgestellt hatte, und seit Herbst 2021 kann der Code als stabil gelten. Der offizielle »Umzug« von der LuaTEX-basierten Version MkIV zu LMTX mit der Veröffentlichung des Quellcodes ist für 2022 geplant, obwohl schon viele Anwender produktiv mit LuaMetaTEX arbeiten.

Das Konzept

In LuaMetaTEX werden Satzmaschine (TEX) und Grafikroutinen (MetaPost) von Lua-Code zusammengehalten und gesteuert.

TEX wird dabei einerseits auf die Knuthsche Satzmaschine reduziert, ohne all die Anbauten für Dateiformate, Schriften und Grafik, andererseits aber zeitgemäßer und flexibler umgesetzt, ohne Rücksicht auf die technischen Beschränkungen der 1980er Jahre. Aus MetaPost wurden die veraltete Schriftverarbeitung sowie Ein- und Ausgabefunktionen entfernt.

Die Verarbeitung von Dateien, Schriften, Grafiken und Farben ist schon in ConTEXt MkIV komplett in Lua geschrieben, so dass große Teile von LuaTEX gar nicht mehr verwendet werden. Diesen Weg geht LuaMetaTEX konsequent weiter.

In der Abkürzung LMTX steht das X für XML, das als Eingabeformat für ConTEXt eine bedeutende Rolle spielt.

Einzelheiten

LuaMetaTEX ist gleichzeitig ein vollwertiger Lua-Interpreter (5.4), erweitert um ein paar praktische Bibliotheken (z. B. `lpeg` und `pplib`), aber ohne JIT und FFI. Die Maschine soll so schlank wie möglich sein und kommt ohne Abhängigkeiten aus. (Die wenigen notwendigen Bibliotheken sind klein, alt und stabil, z. B. `zlib`.) Der vollständige Code wird mit der Distribution verteilt werden, so dass sich das System wahrscheinlich auch auf zukünftigen Systemarchitekturen compilieren lässt.

Der CWEB-Code von LuaTEX wurde zunächst in C-Code konvertiert, der ohne komplexe Infrastruktur auskommt. Artefakte von Pascal und Web wurden entfernt. Die Lesbarkeit des Codes wurde damit deutlich verbessert. Mit Hilfe der Compile-Farm wird laufend getestet, ob sich der Code auf allen Plattformen übersetzen lässt. Dass das schnell und einfach möglich ist, ist für das Projekt essenziell.

Alle betriebssystemabhängigen Funktionen wurden abstrahiert. Alle Dateioperationen laufen über Lua, einschließlich der TEX-Primitiven, die sich auf das Lesen und Schreiben beziehen. Das Gleiche gilt für die Konsolenausgabe.

Die Speicherverwaltung wurde stark verändert – selbst Kleinstrechner haben heute mehr Speicher zur Verfügung als die Computer von damals, wir rechnen nicht mehr mit 8 bit (oder sogar weniger), sondern mit 64 bit, so dass viele Tricks zum Speichersparen heute keinen Sinn mehr ergeben. Das Programm ist trotzdem kleiner als 3 MB und läuft auch auf einem Raspberry Pi 4 recht ordentlich.

Sprachen, Schriften, Marken usw. sind nicht mehr Register-basiert. Schriftinformationen werden nicht mehr im String-Pool abgelegt, was viel ausmacht.

Der weitgehend ungenutzte Code für Richtungen wurde aus dem Kern entfernt und der Rest ins Backend verlagert.

Die Einstellungen für Sprachen wurden effizienter und flexibler implementiert; es wird nur noch wenig in der Formatdatei festgelegt. Die Formatdatei ist damit auch kleiner geworden, obwohl sie nicht mehr komprimiert wird. Das Erstellen des Formats ist stabiler und schneller geworden.

Manche Mathe-Konzepte wurden erweitert, so dass es mehr Kontrolle über die Darstellung gibt. (In diesem Bereich ist im Original-TEX sehr viel hartcodiert, was dann nicht zu allen Schriften passt.)

TEX braucht wenig über Schriften zu »wissen« und wird bei Bedarf mit virtuellen Fonts »gefüttert«, die ihren Namen jetzt wirklich verdienen, denn sie werden nur bei Bedarf im Speicher erzeugt.

Diese Darstellung ist natürlich unvollständig – wer sich für die Einzelheiten interessiert, möge sich die Dokumentation in der ConTEXt-Distribution ansehen.

Makros

Es gibt ein paar Erweiterungen für die Behandlung von Makro-Argumenten und zusätzliche Verzweigungsbefehle, was eine deutlich elegantere Makroprogrammierung ermöglicht.

Eine größere konzeptuelle Änderung sind die Schutzmechanismen: Makros können nicht nur `\protected`, sondern auch `\frozen` sein. Damit werden alle Primitiven und viele grundlegende Makros vor dem versehentlichen Überschreiben geschützt.

Die Versionen `\long` und `\outer` sind dafür verschwunden – was keine große Änderung bedeutet, denn in ConTEXt waren Makros schon immer `\long` und niemals `\outer`, und schon in MkII waren die wenigsten Befehle expandierbar.

In Nodes (vom Zeichen bis zum Absatz) sind mehr Einzelheiten gespeichert und zugänglich, was auch dem Debugging zugute kommt.

Und wie geht es weiter mit LuaTEX?

Natürlich wird LuaTEX weiterhin gepflegt. Schließlich setzt auch ConTEXt MkIV darauf auf, und es ist die Referenz für die Ausgabe von LuaMetaTEX. (LATEX setzt mit LuaHBTEX auf einen Ableger mit eingebauter Schriftverarbeitung.)

Vor allem die LuaTEX-Anwender außerhalb von ConTEXt drängten auf eine stabile Version, es sollte keine Erweiterungen und vor allem keine Änderungen an den Schnittstellen mehr geben.

Hans war zunehmend frustriert von negativen Kommentaren: LuaTEXhabe zu viele Bugs oder sei zu langsam und werde nie in den Distributionen landen, das Handbuch sei zu schlecht, oder auch, das Programm würde in einer kommerziellen Umgebung entwickelt und finanziert. Diese Kommentare sagen wahrscheinlich mehr über ihre Verfasser und ihre Einstellung als über LuaTEX.

Auf der anderen Seite erliegen auch Nicht-ConTEXt-Anwender dem Charme von LuaTEX, und je mehr sie programmieren, desto wichtiger ist es, die Codebasis einzufrieren.

Weil LuaTEX bereits weit verbreitet und fest in TEX live integriert ist, kann nicht mehr viel geändert werden. Es wäre schlechte Werbung für TEX, wenn Low-level-Anwender mit konzeptuellen Änderungen konfrontiert würden.

Für ConTEXt soll die Entwicklung aber nicht stehen bleiben.

Während möglicherweise einmal eine Korrektur oder Erweiterung ihren Weg zu LuaTEX findet, werden die meisten Änderungen in LuaMetaTEX nicht portiert werden; schon jetzt ist die Codebasis zu unterschiedlich. Dafür können die LuaTEX-Benutzer ziemlich sicher sein, dass sich keine Fehler mehr einschleichen werden.