APPENDIX: HOW TEX IS EXTENDED

Plain TEX consists of hundreds of primitive commands, implemented in the TEX program itself, and hundreds more macro definitions. The primitive commands alone provide the fundamental input, output, and typesetting functionality but at too low a level to be convenient for a user to specify the typesetting of a document. The macros turn that base engine into a system that knows the conventions of typesetting. An important group of primitive TEX commands is for defining macros and the mechanisms for calling the new macros. These allow users to augment TEX.

There are many ways augmentation can be done. For instance, one can define new macros to augment plain TEX or redefine existing macros to change the operation of plain TEX. (Eplain is an example of a collection of macros that augment plain TEX, for instance, to add cross-referencing by labels.) In addition, Knut put hooks into TEX to allow the system to work with other programs that TEX doesn't know about (and, largely, that came into existence many years after TEX was finished); for instance, images in various formats can pass through TEX, and TEX can read and write external files (other than the normal input and output).

LATEX, created originally by Leslie Lamport, is a set of macro definitions that reside on top of TEX's primitive commands, while reproducing many of the macro definitions available in plain TEX. LATEX is probably the most common form in which people use TEX, and thousands of macro packages have been written to reside on top of it or to modify its operation, for example, the `url` package that handles the unusual characters in URLs and tries to do sensible line breaking of them, and the `fancyhdr` package that supports nearly arbitrary page header and footer conventions. The AMS-TEX version of TEX is also implemented with macros (on top of plain TEX), as was the later-created AMS-LATEX (on top of LATEX, later merged into LATEX). A user can also add his or her own macros on top of everything mentioned — including making changes to the existing macros.

The TEX ecosystem includes various other levels: `tug.org/levels.html` identifies large collections of TEX-related software; front-ends or editors that provide a development environment, provide sophisticated editing of TEX markup, or provide a graphical user interface; and extended TEX engines that provide new basic functionality at the primitive level. Various packages and engines also allow TEX to be connected to various high level languages instead of the user being forced to program with macro definitions and calls — although some interaction with macros is all but inescapable.

The sustained effort of Hàn Thế Thành to directly produce PDF files from TEX (rather than by conversions from DVI to PostScript to PDF) is an example of how one project dealt with the various levels of TEX. It is also a notable example of how one person became motivated to undertake a significant project and how other people collaborated on it over time, rather analogous to the original TEX project, albeit on a much smaller scale.

- - -

The TEX macro processor used in extending TEX is enormously powerful and flexible (and is a comprehensively documented piece of software).[7,8,9] There are explicit commands in TEX for creating local or global definitions, as well as various other definition variations, such as delayed definition and delayed execution of macro calls. TEX has a rich rather than minimal set of conditional and arithmetic capabilities (some related only to position in typesetting a page). There are also ways to pass information between macros and, more generally, to hold things to be used later during long, complicated sequences of evaluation and computation. These capabilities allow unlimited amounts of new code (programs) for extending or changing TEX to be written in the macro language.

- - -

Historically, there has been an interesting set of pressures around TEX's macro capability. Originally Donald Knuth included only enough macro capability to implement his typesetting interface.

Knuth has made the point that he was designing a typesetting system that he didn't want to make too fancy, i.e., by including a high level language. He has also noted that when he was designing TEX he created some primitive typesetting operations and then created a set of macros for the more complete typesetting environment he wanted. He expanded the original macro capability ("kicking and screaming") when early users, particularly fellow Stanford professor Terry Winograd, wanted to do some fancier things with macros. Knuth's idea was that TEX and its macro capability provided a facility with which different people could develop their own typesetting user interfaces, and this has happened to a large extent, e.g., LATEX, ConTEXt, etc.

That expanded capability allowed users to construct nearly any logic they wanted on top of TEX (although often such add-on logic was awkward to code using macro-type string manipulations). On the one hand, TEX and its macro-implemented derivatives have always been very popular and there have been nonstop macro-based additions for over 30 years. On the other hand, users then and now despair at how annoying coding using macros is, moan about "why Knuth couldn't have included a real programming language within TEX", and otherwise cast aspersions on TEX's macro capability.

It is natural that Knuth used (unfancy) macros to extend TEX rather than high level language constructions. Macros have been and still are traditionally used for user extension of editors and other text processing systems Anyone can understand abbreviations substituting for common phrases of text or sequences of commands. We suggest that TEX's "problem" of not having a programming language to allow user extensions is a primary reason that TEX became so popular and has lasted so long and been built on top of by so many people (and developers, not just users) in so many major ways that in retrospect the decision to have macros and not a programming language can seem unfortunate. RUNOFF, Pub, and Script are gone so people don't complain about extensions to them using macros instead of a programming language.